Development of parametric CAD models for gradient-based aerodynamic shape optimisation

Salvatore Auriemma

A thesis submitted for the Degree of

Doctor of Philosophy

School of Engineering and Materials Science

Queen Mary University of London

April 8, 2021

Statement of originality

I, Salvatore AURIEMMA, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature: Salvatore AURIEMMA
Date: April 8, 2021

Details of collaboration and publications:

- **Optimisation of a U-bend using a CAD-based Adjoint method with differentiated CAD kernel.** Auriemma S., Banovic M., Mykhaskiv O., Legrand H., Mueller J.D., Verstraete T., Walther A. In *ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering* (June 2015), `https://doi.org/10.5281/zenodo.1421284`.

- **Algorithmic Differentiation of the Open CASCADE Technology CAD Kernel and its coupling with an Adjoint CFD Solver.** Banovic M., Mykhaskiv O., Auriemma S., Walther A., Legrand H., Mueller J.D.

Optimization Methods and Software, Volume 33 (2018), https://doi.org/10.1080/10556788.2018.1431235.

- **NURBS-based and Parametric-based Shape Optimisation with Differentiated CAD Kernel.** Mykhaskiv O., Banovic M., Auriemma S., Mohanamuraly P., Walther A., Legrand H., Mueller J.D. In *Computer-Aided Design & Applications* Volume 15(1) (2018) DOI:10.1080/10556788.2018.1431235.

- **Applications of differentiated CAD kernel in gradient-based aerodynamic shape optimization.** Auriemma S., Banovic M., Mykhaskiv O., Walther A., Mueller J.D. In *AIAA Propulsion and Energy Forum* (July 2018), https://doi.org/10.2514/6.2018-4828.

Abstract

Shape optimisation is widely used in industry to improve the performance of the product. When performing aerodynamic analysis with CFD (Computational Fluid Dynamics), gradient-based optimisation methods are normally preferred if the number of design variables is high. These methods require the evaluation of the total derivatives, which can be split into two terms: the flow and the shape derivatives.

While evaluating the flow derivatives with the adjoint CFD method, this thesis demonstrates that the shape derivatives can be calculated with algorithmically differentiated parametric CAD models. The development of such CAD models allows to compute the derivatives exactly and, by utilising the reverse mode variant of algorithmic differentiation, independently of the number of design parameters. This makes the computation of the shape derivatives efficient and robust. The parametrisation of the test-cases (a cooling channel and a compressor stator blade) is defined by intuitive and designer-friendly variables which capture the shape modes which mainly affect the objective function.

The optimised parametric CAD models are compared to reference results. These results are set as the optimal shapes given by parametrisations with refined design space. The reference results of the cooling channel are identified in the literature. For the blade test-case, the design space of the parametric-based CAD model is enlarged (almost quadrupled). The optimised shape obtained with the parametric-based design is able to reproduce the same design modes provided by the enlarged design space.

The fit of the assembly constraints of the blade's test-case (four mounting bolts) during the flow optimisation has never been demonstrated. This is due to the arduous identification of a differentiable assembly constraints' function. This thesis demonstrates that an approach based on the detection of a signed distance between the blade and the bolts succeeds in fitting the assembly constraints.

Acknowledgments

The achievement of this academic title is a dream that becomes true. It arrives at the end of a long series of experiences, unexpected problems, emotions, marvelous people that I have met along the last years.
I thank Prof. Dr. Jens Dominik Mueller for his passionate and precise supervision. He was always keen to follow the several steps of my research while still allowing sufficient freedom to my day-by-day investigations. This thesis would have never been like it is today without his support. I want also to mention my industrial supervisors at Open CASCADE, Mr. Hervé Legrand and Mr. Sergey Slyadnev. Their extensive experience in CAD development was crucial to let me achieve the main goals of my research. I thank Mr. Mikhail Kazakov and Mr. Daniel Brunier-Coulin for the warm welcome that they reserved to me since my arrival at Open CASCADE. Last but not least, my two examiners, Prof. Dr. Cecil Armstrong and Dr. Yi Sui. Their meticulous review allowed me to see my research from a different point of view and to effectively improve my thesis. It was an honour to have my manuscript approved by such a prestigious panel.
This thesis was conducted within IODA project - Industrial Optimal Design using Adjoint CFD. IODA is a "Marie Sklodowska-Curie Innovative Training Network" funded by European Commission (Grant Agreement No. 642959). Therefore, it is mandatory to mention also the European Commission which provided the funds for my scholarship.
I dedicate this thesis to my family, in particular my mother and my father. They gave me a strong support which helped me to sort out several issues. This is also true for my friends, who were always from my side during these beautiful years.
I want to finish with a sentence which was told to me by my former boss when I resigned for joining this PhD:"good luck, and bear in mind that normally life experiences are as beautiful as difficult". It was true, I confirm that this experience was outstanding.

Contents

Nomenclature

EA	Evolutionary Algorithms
GA	Genetic Algorithms
J	Cost function
α	Design parameters
X_V	Grid mesh points
X_S	Surface mesh points
FD	Finite Difference
CV	Complex variable
AD	Algorithmic Differentiation
TLM	Trace-less forward Mode (AD)
TM	Trace mode for (AD)
NURBS	Non Uniform Rational B-spline
HH	Hicks-Henne functions
FFD	Free-Form Deformation
OCCT	Open CASCADE Technology
ADOL-C	Automatic Differentiation by OverLoading in C++
RM	Reverse Mode

SVD	Singular Value Decomposition
MDO	Multidisciplinary optimisation
DoF	Degree of Freedom
RBF	Radial Basis Functions
$P_{i,j}$	NURBS control points
$W_{i,j}$	NURBS weights
p, q	degrees of the NURBS surface in u and v direction
NSPCC	NURBS-based parametrisation with complex constraints
O-O	Operator Overloading
PARSEC	Parametric section
κ	Curvature of the B-spline curve in a point
BFGS	Broyden–Fletcher–Goldfarb–Shanno
s	Arc lenght of the pathline
h	Step size of the Taylor test
n, m	Number of design parameters and of cost function, respectively
STAMPS	In-house discrete Adjoint Solver
IDW	Inverse distance weighting
API	Application Programming Interface
U_1, U_2	U-bend main design modes
SQP	Sequential Quadratic Programming
NLP	Non Linear optimisation Problem

The sum of intelligence on the planet is a constant; the population is growing.

Cole's axiom

Chapter 1

Introduction

1.1 Background

During the last decades, the main objectives pursued by industries in sectors such as automotive and aeronautics are represented by the reduction of the environmental impact (in terms of pollutants and of noise), the improvement of the product safety, the reduction of the costs and of the time necessary to commercialise a product. One of the key features of the strategy defined by the manufacturers to achieve these objectives is to constantly review and optimise the industrial workflow, i.e. make it more efficient.

The manufacturing process is now being innovated by the massive application of the digital technologies to the production plant (known as *Industry 4.0* [1]). The current design process, which extensively employs computers to define the geometry of the product with Computer Aided Design (CAD) software and to analyse the aerodynamic performance by using Computational Fluid Dynamics (CFD), is also changing. Past decades have seen the development of optimisation algorithms. These algorithms try to improve the performance of a design by changing the parameters controlling its shape. The new set of parameters generates an updated shape which is aerodynamically tested with CFD calculations. These calculations are computationally expensive and require from hours to days on computer clusters [2, 3].

Shape optimisation can be done by utilising evolutionary or genetic algorithms, which are defined as gradient-free methods because they do not utilise the derivatives to solve the optimisation problem. Gradient-based methods are preferred to gradient-free ones because they normally employ a reduced number of CFD evaluations [4].

One of the important ingredients for the routine application of gradient-based

methods in engineering design is represented by the definition of a mature and efficient optimisation framework, which includes the CAD software. Such a framework is used in this thesis to optimise the geometry of the test cases with a limited number of CFD calculations.

1.2 Manual design loop

Over the past decades, the introduction of computers as design tools has massively changed engineering design. Before this introduction, it was a common practice in industry to build prototypes whose performance was verified with experimental tests. Changes to the geometry were applied based on the analysis of the results of these tests. This process was repeated until a satisfactory performance was identified.

The utilisation of Computer Aided Engineering (CAE) software has partially

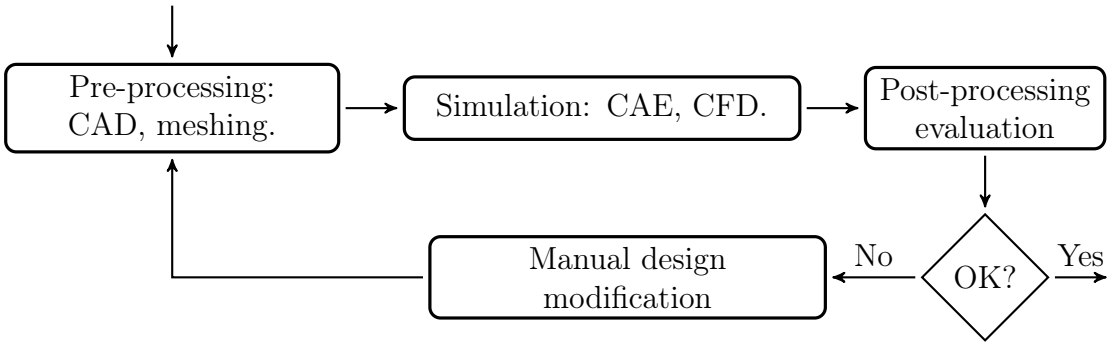


Figure 1.1: Manual design loop [5].

replaced some of the experimental tests with virtual ones. This has not changed the iterative nature of the process but has allowed to reduce the time and the costs of the design workflow. Moreover, by using this design loop, the virtual model of the geometry is generated manually in the CAD software such that the design constraints can be straightforwardly incorporated.

The main limitation of this process (know as "manual design loop", Fig. 1.1) is represented by the lack of a design parametrisation, i.e. the definition of how the design variables affect the shape, and of a robust way to rebuild a consistent geometry with a different set of design variables. Also, the manual intervention of the designer to change the shape is labour-intensive and only small design spaces, i.e. a limited set of design solutions, can be considered. The manual design loop

therefore does not allow to explore all the possible designs. Another important aspect to be considered is that human intervention relies on intuition. Typically the user does not take into account a set of designs due to constraints which arise from his experience. On the other hand, the innovation might be found among those designs that were discarded.
Numerical optimimisation is investigated by the scientific community to avoid the aforementioned drawbacks.

1.3 Numerical Optimisation

The first step of the numerical design loop (Fig. 1.2) is represented by the definition of the parametrisation, which defines the design space to be explored and is ideally able to respect the design constraints. After having analysed the performance of the shape with CAE simulations, the optimiser selects a set of design variables that generate a new shape with hopefully improved performance. The design loop continues by executing another simulation and the optimisation stops when the optimiser identifies the optimal shape (i.e. the optimisation converges).
The order of the derivatives of the objective function (i.e. the scalar cost function

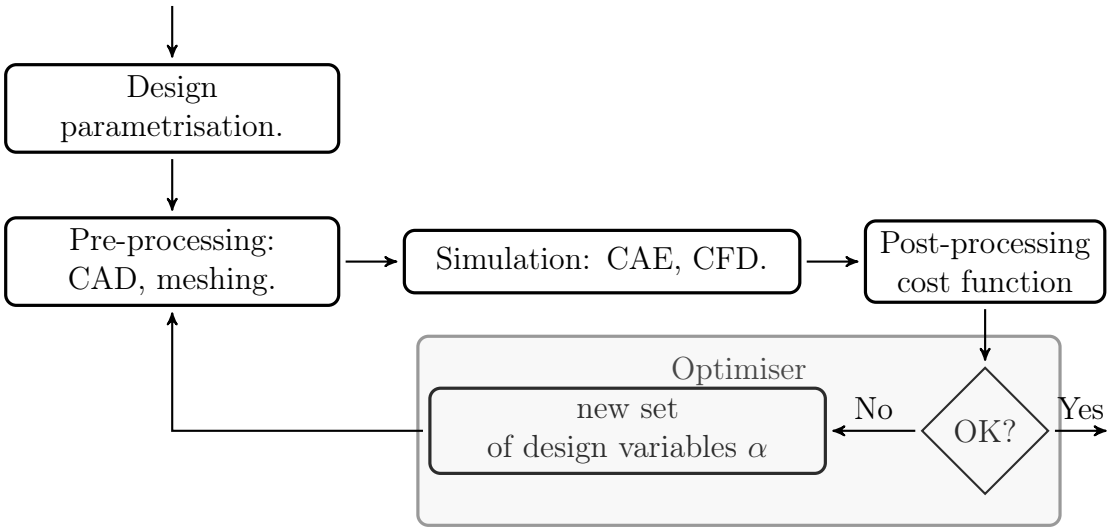


Figure 1.2: Design loop with numerical optimisation [5].

to be optimised) used to perform the optimisation is utilised to classify the optimisation methods. Zero-order methods [6, 7], also known as gradient-free methods, consider only the value of the objective function in the search for the optimum

whereas gradient-based methods [8–11] employ first and second order derivatives. By eschewing the utilisation of the derivatives, gradient-free methods globally explore the design space such that the risk of stumbling in a local minimum/maximum of the objective function (where the derivatives are nullified) is avoided. On the other hand, after having determined a promising region of the design space, gradient-free methods do not guarantee the identification of the local optimum.
Gradient-free methods can be mainly divided in two branches: (i) stochastic and (ii) meta-models. The first ones explore the design space with stochastic sampling. An initial global exploration allows to identify the promising areas of the design space. These areas are then subject to further investigation to identify the optimal set of design variables. Typical stochastic methods are the Evolutionary Algorithms (EA) [12] and the Genetic Algorithms (GA) [13]. These methods are commonly used when the computational cost of an evaluation is low (e.g. linear structural optimisation). The meta-model methods [14] are often combined with stochastic ones. These methods fit a curve/surface (response surface, RS) through the sampled points. After the identification of the optimum, the RS is updated with new samples.
The number of evaluations of the objective function required by stochastic methods scales strongly with the size of the design space and can be very high when the design space consists of more then 50-100 parameters [15]. If the parametrisation consists of hundreds of parameters, gradient-based methods are considered computationally more convenient w.r.t. gradient-free methods [16], as only a monodimensional path (defined by the derivative information) through the design space needs to be traced. This has also been demonstrated by Yu et al. [17], who verified that, for a wing test case, gradient-based methods are at least four times less expensive w.r.t. gradient-free ones to reach the convergence.
This research aims to perform the aerodynamic shape optimisation of industrial test cases by utilising gradient-based optimisation methods.

1.4 Gradient-based aerodynamic shape optimisation

Gradient-based methods (Fig. 1.3) require the calculation of the derivatives of the cost function J w.r.t. the design parameters of the optimisation α_i, i.e. the total derivative $\frac{dJ}{d\alpha}$. The calculation of the derivatives can be obtained by using several methods, which are explained in Sec. 3.1. A first method is the "exact differentiation", that computes the derivatives analytically, i.e. by hand. The

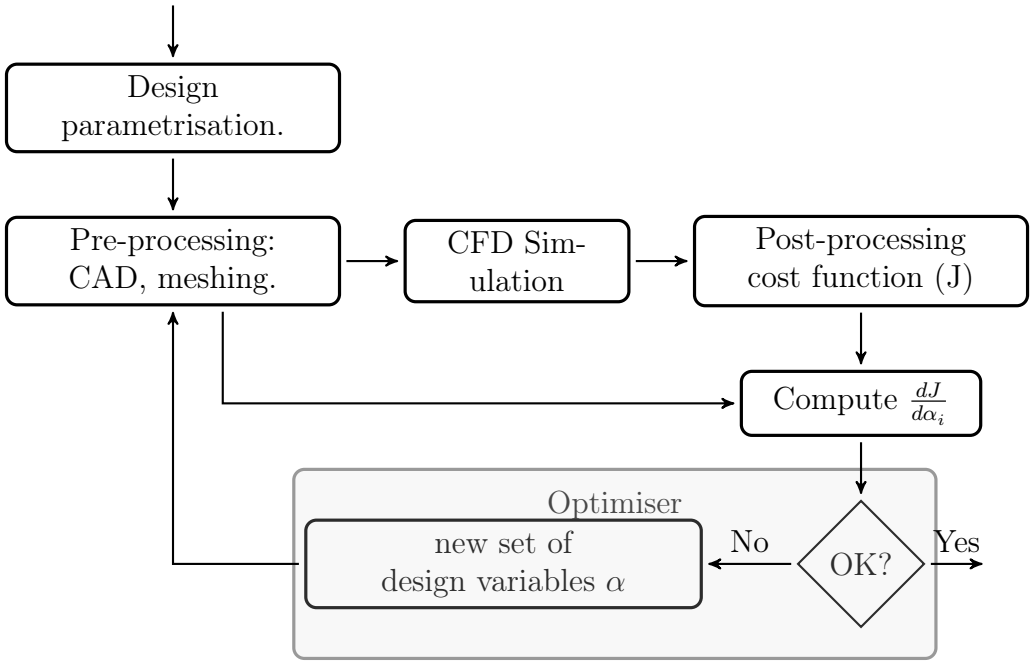


Figure 1.3: Numerical optimisation with gradient-based optimiser [5].

knowledge of the equations implemented within the software that computes the objective function is therefore necessary. Moreover, the software that computes the derivatives has to be implemented manually, which makes this method difficult to utilise for complex equations. The Finite Difference (FD) method does not require the knowledge of the equations or of the source code. Its computational cost scales with the number of design variables and it does not exactly compute the value of the derivatives. The complex variable (CV) method, which needs the source software to be implemented, can provide an exact calculation of the derivatives. However, as for FD, the computational cost of CV scales with the number of design variables.

Another possible solution to compute the derivatives of the objective function w.r.t. the design variables is the algorithmic differentiation (AD) technique [18]. Given the source code of the program to be differentiated (i.e. the primal), this technique (Sec. 3.2) relies on the main concept that one can differentiate each statement of the primal code exactly, and assemble the complete derivative using the chain rule of calculus. To apply it to a certain program, one has to integrate an AD tool into its sources. An AD tool can be applied in forward and reverse mode, both of them providing an exact evaluation of the derivatives. The forward mode computes the derivatives of the objective functions w.r.t. a single design

parameter at once. This implies that, in order to complete the computation of the derivatives, the objective functions have to be evaluated n times, if n is the number of design parameters. The computational cost of this mode, therefore, scales with the number of parameters and is independent on the number of objective functions. On the other hand, the reverse/adjoint mode calculates the derivatives of a single cost function w.r.t. the full set of design parameters at once such that the computational cost increases with the number of objective functions but is constant in the number of design variables.
In aerodynamic shape optimisation, the number of design variables is much bigger then the number of objective functions. This research work employs the reverse mode of AD to compute the total derivatives.

1.4.1 Total derivatives

The computation of the total derivatives is typically split into two parts, i) the computation of the derivative of the objective function w.r.t. to the mesh surface coordinate (the flow derivative), and ii) the derivative of the mesh surface coordinate w.r.t. to the design parameters (the shape derivative). Here, the flow derivatives are computed with the efficient Adjoint CFD method (Sec. 6.1). The calculation of the shape derivatives depends on the parametrisation approach chosen to control the geometry during the flow optimisation.
In engineering design, it is common practice to define the geometry with the features provided by the CAD software. This software allows to determine an internal representation of the geometry, the CAD model, that embeds geometric constraints such as the constant axial chord of an airfoil profile. This CAD model is used for further analysis with design and manufacturing tools.
The parametrisation approaches used in shape optimisation are distinguished in CAD-free and CAD-based parametrisations. CAD-free parametrisations (Sec. 2.2) allow to straightforwardly calculate the shape derivatives. This makes easy their integration into the design chain. The main drawback of CAD-free parametrisations is that the optimal shape is not produced in CAD format. The transformation to a format digestible for CAD software is therefore necessary. The parametrisation which control the CAD model considered to re-approximate the optimised shape has to be sufficiently rich in order to reproduce all the relevant geometric features. On the other hand, the designer prefers to retain engineering parameters and to work with the coarse design spaces used for manual design.

Hence, despite the support of specific software[1], relevant geometric features are often not reproduced by the final CAD model.
In this thesis, CAD-based parametrisations are utilised to control the geometry. These parametrisations allow to avoid the limitations presented by CAD-free ones.

1.5 CAD-based parametrisations

CAD-based parametrisations utilise the parameters controlling the CAD model to define the design space of the optimisation. There are two types of CAD-based parametrisations: parametric-based designs and NURBS-based ones, which utilise Non Uniform Rational B-spline (NURBS) to parametrise the geometry.
Parametric-based designs (Subsec. 2.3.1) control the geometry with engineering parameters normally used by designers to conceptualise the component such as thickness, camber of an aircraft wing or a turbomachinery blade. NURBS-based parametrisations (Subsec. 2.3.2) are based on the boundary rapresentation (BRep) of the CAD model, which is the most prevalent form to interchange the geometry between modelling tools/disciplines. These parametrisations provide a richer design space w.r.t. parametric-based ones and are normally used to explore new design solutions.
In this thesis, parametric-based designs are chosen to control the geometry of the test cases. This parametrisation approach suits the typical design workflow used in industry. First, it allows the utilisation of the parameters normally handled by the designer during the conception phase of the product. Secondly, the design constraints such as the thickness distribution of an airfoil profile can be imposed directly while defining the parametrisation. The assembly constraints, which determine how components in an assembly fit together, can also be considered if a differentiable assembly constraint function is determined. This is not an easy task, which is currently subject of research within the scientific community [19].

1.5.1 Calculation of the shape derivatives

In literature, several approaches have been used to compute the shape derivatives. These derivatives are not provided by commercial CAD systems such as CATIA[2] and Siemens NX[3]. The only approach which allows to compute the derivatives by

[1] `http://www.spaceclaim.com/en/default.aspx`
[2] `https://www.3ds.com/products-services/catia/products/`
[3] `https://www.plm.automation.siemens.com/global/en/products/nx/`

employing such CAD systems is the FD method. This approach has been investigated by Robinson et al. with several works [19–21]. The key features of the FD method have been introduced in Subsec. 1.4.1. Subsec. 2.3.1 provides further details about its main benefits and drawbacks. Another possible approach is to develop an in-house CAD kernel tailored to a specific application field. J. Grasel et al. [22] have implemented a geometric kernel customised to turbomachinery blade with the parameters normally used by designers such as radii of trailing and leading edges. Verstraete has implemented a CAD library called *CADO* for parametrising turbomachinery components such as blades and cooling channels [23]. The small dimensions of the code make it adaptable to the utilisation of several techniques for the computation of the shape derivatives such as algorithmic differentiation [24] and the complex step method [25]. On the other hand, such a small library can not be used to parametrise other types of test cases. This makes prohibitive the extension of this library to other application fields.

Dannenhoffer and Haimes [26] use the open-source CAD kernel **Open CASCADE Technology (OCCT[4])** as a geometric engine. They build the CAD model as a combination (i.e. boolean operations) of analytic parts (i.e. CAD primitives), for which the derivatives can be formulated. For any part other than simple ones (such as circles and cylinders defined by origins, radii and axes), the analytic differentiation is not straightforward. For these "complex" parts, the differentiation is obtained by employing the FD method. It has still to be demonstrated the applicability of analytic differentiation to the complex parts implemented by OCCT. Moreover, the CAD kernel is just a part of the CAD system. Further algorithms which are implemented by using the functionalities provided by the CAD kernel (e.g. the constraint solver) should also be differentiated by hand. This differentiation would further complicate the utilisation of this type of approach.

Most often the baseline geometry to be optimised is not equipped with a shape parametrisation, or the parametrisation is arising from a particular design workflow and does not offer design variables that are suitable for optimisation. This implies that, after the definition of a design space with parameters tailored for shape optimisation, the values of these parameters that "best fit" the existent baseline geometry have to be determined. Agarwal [27] fits the surface mesh of a climate duct with a parametric CAD model by solving a gradient-based optimisation. The objective function of the optimisation is the "total distance" between the surface mesh and the parametric CAD model and the derivatives of the objective function w.r.t. the CAD parameters are computed by using the FD method.

[4] `https://www.opencascade.com/`

1.6 Motivation of the research work

The application of Algorithmic Differentiation to the sources of a complete CAD system is an novel approach to compute the shape derivatives, which has never been demonstrated before. A differentiated CAD system allows the definition of automatically differentiated CAD models. Such CAD models can be used to define a fully differentiated design chain with CAD in the loop.
The investigation of this approach is enticing for the scientific community for several reasons. First, the derivatives are computed robustly and exactly. Secondly, the derivatives can be computed with the efficient reverse mode, which makes the computational cost independent from the number of design variables that control the CAD model. Then, the differentiated CAD kernel can be used to parametrise a wide range of industrial components. Finally, the manufacturing constraints included the assembly ones can be imposed by utilising the CAD features presented by the CAD system.
In this thesis, which is inserted in a broader research program focused on gradient-based aerodynamic shape optimisation ongoing within the European framework IODA[5] (Industrial Optimal Design using Adjoint CFD), the differentiated version of the open source CAD library **OCCT** is utilised to develop automatically differentiated parametric CAD models. This library is one of the main CAD systems available on the market and is the only one open source. This has made this library the most suitable to the application of the AD tool ADOL-C.

1.7 Thesis objectives

The primary objectives of this thesis are:

1. Design of parametrisations utilising the differentiated version of the open source CAD library **OCCT**. In particular, the CAD models should be controlled by parameters that are normally used by the designer during the conception phase of the component.

2. Design an algorithm that refits the existent baseline geometry of a test case with the parametrisation implemented to perform the aerodynamic shape optimisation by utilising the CAD kernel OCCT differentiated in reverse mode.

[5] `https://ioda.sems.qmul.ac.uk/`

3. Perform the CAD-based aerodynamic shape optimisation of the test cases (a cooling channel and a compressor stator blade) and verify that the parametrisation proposed in this thesis is able to reproduce the same design modes obtained with reference parametrisations.

4. Impose manufacturing constraints during the flow optimisation. In particular, the constraints that have to be considered are the following:

   - constraints imposed by the parametrisation by construction (e.g. constant axial chord of a blade),
   - constraints imposed by defining boundaries to the design variables within which find the best set of parameters (e.g. lower/upper thickness distribution of the blade not to be exceeded).

5. Investigate the possible approaches that can be used to respect the assembly constraints during the flow optimisation and determine a differentiable assembly constraint function.

1.8 Outline

This section presents the outline of this thesis, which consists of 6 main chapters. From Chapter 2 to chapter 5 the main ingredients necessary to implement the automatically differentiated CAD models are investigated. Chapter 2 studies the main parametrisation approaches (both "CAD-free" and "CAD-based" ones) used in gradient-based shape optimisation and explains the key features of parametric-based design. Chapter 3 introduces the fundamental concepts of AD and presents the differentiation of **OCCT**, which is obtained by using the AD tool ADOL-C ("Automatic Differentiation by OverLoading in C++"). Chapter 4 explains how the differentiated CAD kernel allows to implement an algorithm that refits the existent baseline geometry of an airfoil with the parametrisation developed with the differentiated OCCT. Chapter 5 gives details about the implementation of the feature tree of the automatically differentiated parametric CAD models (a cooling channel and a compressor stator blade). This chapter also provides the comparison berween the shape derivatives computed with the differentiated **OCCT** and the ones calculated with the FD method.

Chapter 6 and chapter 7 focus on the application of the automatically differentiated CAD models in aerodynamic shape optimisation. Chapter 6 introduces

to the CAD-based shape optimisation framework utilised in this research, which chains the differentiated **OCCT** to STAMPS, the in-house adjoint CFD solver developed at Queen Mary University of London. This framework is applied to aerodynamically optimise the first test case studied in this thesis: a cooling channel for high-pressure turbine blades (i.e. the U-bend). The shape optimisation results are compared to reference ones identified in literature.

Chapter 7 studies the possible approaches that can be used to impose the assembly constraints during the flow optimisation. The chosen approach allows to perform the shape optimisation of the compressor stator blade while embedding all the manufacturing constraints. In literature, it is not possible to find any application of such test case which respects also the assembly constraints. The optimisation results given by the automatically differentiated CAD model proposed in this research are therefore compared to the ones provided by the refined parametrisation implemented by the author.

Chapter 8 provides the conclusions and the further work.

Chapter 2

Parametrisation approaches in aerodynamic shape optimisation

The definition of the parameters controlling the geometry is one of the major challenges in shape optimisation. The range assumed by the parameters define the design space. Therefore, a careful selection of these parameters is necessary in order to identify the shape with best performance. This chapter introduces the main parametrisation approaches used in shape optimisation and provides a detailed literature review. Sec. 2.1 explains which are the main features that characterise a proper parametrisation. Sec. 2.2 and Sec. 2.3 give an extensive literature review about CAD-free and CAD-based parametrisations, respectively.

2.1 Introduction

As explained in Sec. 1.2, the parametrisation is the definition of how the parameters affect the geometry. The selection of a proper parametrisation is necessary to let the geometry assume a wide variety of shapes during the flow optimisation. This is a key ingredient to identify the shape with best performance. The features that characterise a proper parametrisation have been summarised by Xu [28]:

- provide a design space with a broad set of smooth deformations, i.e. broad enough to let the identification of all the relevant modes while guaranteeing the smoothness of the optimised geometry.
- Implement the main functionalities provided by CAD systems such as the definition of intuitive parameters to control the geometry and the storage of the model in standard CAD formats (e.g. STEP and IGES). This allows to

share the geometry with various tools in the design, analysis and production workflow.

- Allow the imposition of geometric constraints: assembly, manufacturability, etc.
- Require limited computational resources for the integration in the design loop.
- Provide the calculation of the shape derivatives (Subsec. 1.4.1).

The parametrisation approaches can be divided mainly into two branches: CAD-free and CAD-based ones. CAD-based parametrisations utilise CAD models to define the design space. On the other hand, CAD-free methods eschew the utilisation of CAD features to set up the parametrisation. This allows to avoid the calculation of the derivatives of the CAD model. A detailed literature review of CAD-free and CAD-based parametrisations is given in Sec. 2.2 and Sec. 2.3.

2.2 CAD-free parametrisations

In this section, the author investigates the CAD-free parametrisations mainly used in shape optimisation. The parametrisations investigated are: node-based [29], Free-From Deformation (FFD) [30], Radial Basis Functions (RBF) [31], Hick-Henne (HH) bump functions [32], a method based on Singular Value Decomposition (SVD) [33] and the polynomial and spline representations [34].

Node-based parametrisations define as design variables the coordinates of the mesh points. This is the richest possible design space and, theoretically, it allows to represent any shape. This approach has been used by several authors such as Jameson [29] Campbell [35] and Wu [36]. The drawbacks of this approach are mainly two. Firstly, in order to guarantee a manufacturable optimised geometry, additional smoothing is needed [37]. The motivation is that irregular gradients cause high-frequency oscillatory modes [38]. These modes are not detected by the flow solver and, thus, not correctly cancelled by the optimiser [28]. Two are the possible solutions to this problem. One is the smoothing/regularisation of the gradients [39]. Another solution is to smooth the displacement rather then the gradient [40–42]. Secondly, the optimised mesh has to be transformed to CAD format [43]. This transformation is very difficult to obtain manually also for an experienced user of CAD

software and could implicate the loss of details with consequent reduction of the aerodynamic performance.

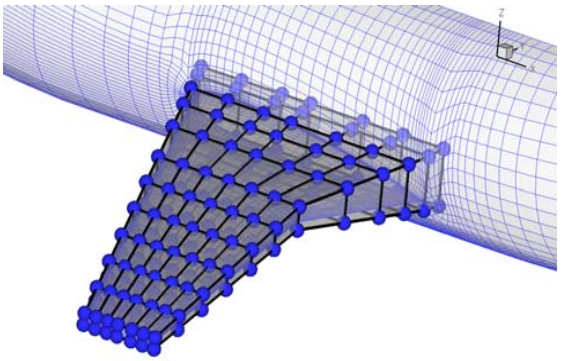


Figure 2.1: Possible parametrisation of an aircraft wing using the FFD approach [44].

Free-Form Deformation (FFD) method is another approach that has been firstly applied by Barr [45] and Sederberg [30]. FFD methods are based on a surface morphing technique that has been originally used in computer graphics for animation motion and deformation of solid models. This type of parametrisation can be visualised as embedding the spatial coordinates defining a geometry inside a flexible volume defined by a volume spline. The shape of the volume spline can be changed by modifying its control points. Fig. 2.1 shows the application of the FFD approach used to parametrise the wing of an aircraft [44]. One of the main benefits of FFD is that it provides an algebraic scheme to perturb the mesh points within the control volume, as opposed to the node-based parametrisations, where only the surface is perturbed. As consequence, the calculation of the derivatives of the volume mesh points w.r.t. the design variables can be performed analytically, as reported by Samareh [46] and Yamazaki et al. [47]. Another advantage of FFD is that, since it can be used for volume-based deformation, it allows the execution also of structural deformations. Widhalm et al. [48] optimised the M6 wing. Yamazaki et al. [47] showed the application of FFD to parametrise 2D aerofoils. However, it is hard to find in literature the application of FFD to complex geometries [49]. Complex geometries would require the definition of a multitude of FFD boxes to deform some parts, and not others. To keep the geometry consistent is a major challenge and current topic of research. Also, as for the node-based approach, the optimised shape is a deformed mesh which has to be transformed in a format digestible for CAD systems.

Radial Basis Functions (RBF) [50] can also be employed as morphing technique. The Radial Basis Function is a function that varies with distance from its origin. This parametrisation approach sums to a baseline function a set of RBF functions such that every RBF function is weighted by a coefficient. One has to solve a linear system of equations to find the weights of these functions such as to interpolate point values. For every point to be interpolated, an extra RBF has to be introduced. The main advantage of RBF parametrisation is that, due to its construction, the deformation is smooth [50–52]. Also, this avoids the need to implement a separate volume mesh deformation. The main challange of RBF approach is that it presents several difficulties to parametrise complex geometries with movable and fixed parts. Since the RBF method is global (i.e. each RBF function has global reach from the origin), every mesh point that is fixed is pinned down with an interpolating RBF. This leads to very large system of equations to compute the weights and, therefore, to high computational costs and memory requirements. To reduce these costs, Gagliardi et al. [53] utilise a two-step RBF interpolation that use the Sparse Approximate Inverse (SPAI) preconditioner and the Fast Multiple Method (FMM) to optimise a turbomachinery row. However, an approach that allows to substantially reduce the computational requirements of RBF parametrisation for a wide range of geometries has still to be identified. Also, RBF parametrisation still provides the optimised shape as a mesh.

Lattice-based methods [32, 54] allow to reduce the number of design variables compared to the node-based ones. These methods can be defined by summing to a baseline function a set of shape function. Per every shape function, a weighting coefficient is considered. These coefficients are set as the design parameters. The shape functions could be Hicks-Henne functions [32], Wagner functions [55], Legendre functions [54], Bernstein functions [54]. Among them, the 'Hicks-Henne' (HH) are the most used. These basis functions are defined as in Eq. 2.1:

$$f_i = [sin(\pi x^{\beta_i})]^w \qquad 0 \leq x \leq 1 \tag{2.1}$$

with:

$$\beta_i = \frac{ln(0.5)}{lnx_{Mi}} \tag{2.2}$$

where x_{Mi} controls the position of the maximum peak point for the i-th bump function and w controls the width of the bump function. The key feature of this parametrisation approach is that the HH basis function is defined on a topologically rectangular grid. This aspect of HH allows to use functions that span the whole rectangular domain and allows to define nearly-orthogonal chord and span-wise deformation functions.

The HH functions have been used by several authors as parametrisation approach in aerodynamic shape optimisation [56–59]. Most of the applications are related to planar geometries, mainly airfoils. Nakayama et al. [58] modeled an airfoil controlled by 71 design parameters. Kim and Nakahashi [60] considered 37 design variables and carried out a high-lift device optimisation with the unstructured adjoint method. Also Sung [61] optimised an airfoil profile by using HH parametrisation. The author highlights that it is difficult to find in literature application of HH parametrisation to complex 3D geometries.

The approach based on Singular Value Decomposition (SVD) is used by Poole et al. [33,62,63]. They store a high number of airfoil profiles (e.g. thousands) in a library and analyse this library by using SVD in order to pick up higher-frequency modes. These modes are then linearly combined by using coefficients that are then utilised as design parameters of the optimisation. This approach allows to reduce the total number of design variables w.r.t. node-based parametrisation. Also, the airfoils are selected in literature based on their aerodynamic performance. A combination of these airfoil profiles should allow the identification of an optimal shape with high aerodynamic performance.

Bezier, B-spline and NURBS representations are other approaches that allow to reduce the number of design variables w.r.t. the node-based ones (Subsec. 2.3.2). Bezier curves/surfaces are normally used when dealing with low order curves/surfaces controlled by a reduced number of control points. The main drawback of Bezier curves is that the design parameter (i.e. control point of the Bezier curve) does not have local control on the curve. Indeed, by changing a control point position, all the curve changes its shape. This limitation is overcome by using piecewise polynomials, i.e. B-spline curves/-

surfaces. The main drawback that still characterises this type of curves/-surfaces is that they are polynomial based. This implies that they cannot implicitly define conic shapes (i.e. circles, ellipses etc.). NURBS allow the definition of conic shapes. Lepine et al. [64, 65] use NURBS to parametrise airfoils. The optimisation results show that NURBS parametrisation is able to provide a smoother profile w.r.t. the one obtained by parametrising the geometry with the HH approach. One of the main drawbacks of this approach is that the geometric derivatives are computed using the FD, which implies that the computational costs scale with the size of the design space. Painchaud-Ouellet et al. [66] parametrise an airfoil while considering the thickness constraint. The design variables are the control points and the weights of the NURBS curve. The work is only related to the parametrisation of the 2D airfoil and there is no link back to CAD. Also, the usage of FD for calculating the derivatives could cause excessive computational costs.
Summarising, several authors use NURBS to perform CAD-free shape optimisation. The main drawbacks presented by all the aforementioned works are mainly two. First, the derivatives are calculated using FD. This makes the computational costs, which scale with the number of design parameters, not negligible due to the high number of variables normally used to parametrise the geometry with NURBS.
Secondly, none of these works provides a link back to the CAD system. Importing the optimised shape back to CAD for further analysis or manufacturing is a complex task. The manual transcription of key features could oversimplify the shape such that critical details could be lost. Over the years, "back-to-CAD" software such as SpaceClaim[1] have been developed to support the designer in this task. As practice shows[2], such reverse engineering is rarely fully automatic.

2.3 CAD-based parametrisations

CAD-based parametrisations utilise the parameters controlling the CAD model as design variables such that the optimised shape (i.e. the optimised CAD model) can be shared with tools for further analyses in the design and manufacturing workflow. There are two possible ways of parametrising a geometry by using the CAD features. The first approach, the parametric-based design (Subsec. 2.3.1), directly

[1] `http://www.spaceclaim.com/en/default.aspx`
[2] `https://www.youtube.com/watch?v=sesc4ZrZ9fo`

parametrises the geometry with engineering parameters (i.e. intuitive and user-friendly parameters such as thickness distribution, camber-line curve etc.). The second one, the NURBS-based design, relies on NURBS curves/surfaces (Subsec. 2.3.2).

2.3.1 Parametric-based design

Parametric-based design (also known as "parametric design") relies on the implementation of parametric CAD models, which are computer representations of the shape with some geometric properties that are fixed and others that can change. The fixed properties are defined as constraints whereas the changeable ones are defined as the parameters of the model [67]. The CAD system is the environment where the parametric CAD model is implemented. Nowadays, the CAD systems mainly used in parametric design are CATIA[3], Siemens NX[4], Solid Edge[5] and PTC Creo[6]. These tools allow the definition of a CAD model by using a user-friendly Graphical User Interface (GUI).

In Fig. 2.2, the parametric CAD model of a screw nut is shown in *Shaper*[7], the CAD system developed by Open CASCADE, Commissariat à l'énergie atomique et aux énergies alternatives (CEA), and Électricité de France (EDF). The parametric CAD modeling foundation has been established by Shah [68] and Roller [69] and supported by the work of Anderl and Mendgen [70]. Many authors have studied parametric CAD modeling in terms of the technology, its mathematical basis, and its benefits within the product development process [70–73]. Parametric models are usually defined as "feature-based" since they consist of various features. In a CAD model, every *feature* encapsulates the engineering significance of portions of the geometry of a part or assembly [68].

The complete set of features is listed in the feature tree, which also considers the constraints between connected features. If an individual feature is modified or changed, the overall feature tree is re-evaluated. Organisations normally use parametric CAD models when designing families of products that include slight variations on a core design. This allows the designer to define the baseline 3D model of the component just once and then update this baseline in order to retrieve several different products. In other words, parametric-based design eases

[3] https://www.3ds.com/products-services/catia/products/

[4] https://www.plm.automation.siemens.com/global/en/products/nx/

[5] https://solidedge.siemens.com/en/solutions/products/3d-design/synchronous-technology/

[6] https://www.ptc.com/en/products/cad/creo

[7] https://docs.salome-platform.org/latest/gui/SHAPER/index.html

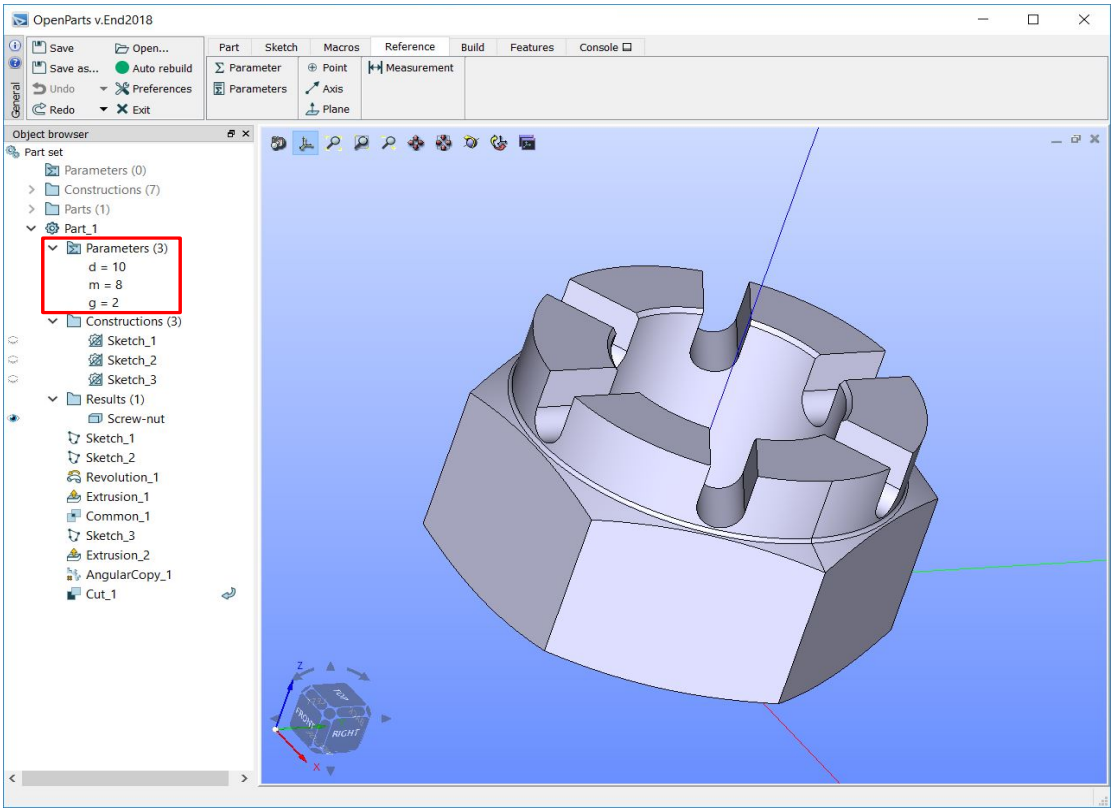


Figure 2.2: Parametric CAD modeling of a screw nut in the CAD system "Shaper". The controlling parameters (highlighted by the red box) of the model are defined in the tree on the left.

the design of products that will need to be modified or iterated on a regular basis. An example of parameter's update of the CAD model shown in Fig. 2.2 is presented in Fig. 2.3.

An important advantage of using parametric CAD modeling as parametrisation approach is that the constraints (e.g. constant axial chord in an airfoil profile) are defined directly in the design space. This simplifies the integration of the aerodynamic shape optimisation chain with manufacturing processes. On the other hand, the definition of the design parameters requires an extensive knowledge of the flow properties of the test case, e.g. how the fluid evolves in a turbomachinery row.

Literature review

Several groups of research have used parametric-based design in shape optimisation. In particular, mainly three types of approaches can be found in literature:

1. development of geometric kernels implemented to construct the geometry of a specific test case such as the wing of an aircraft.

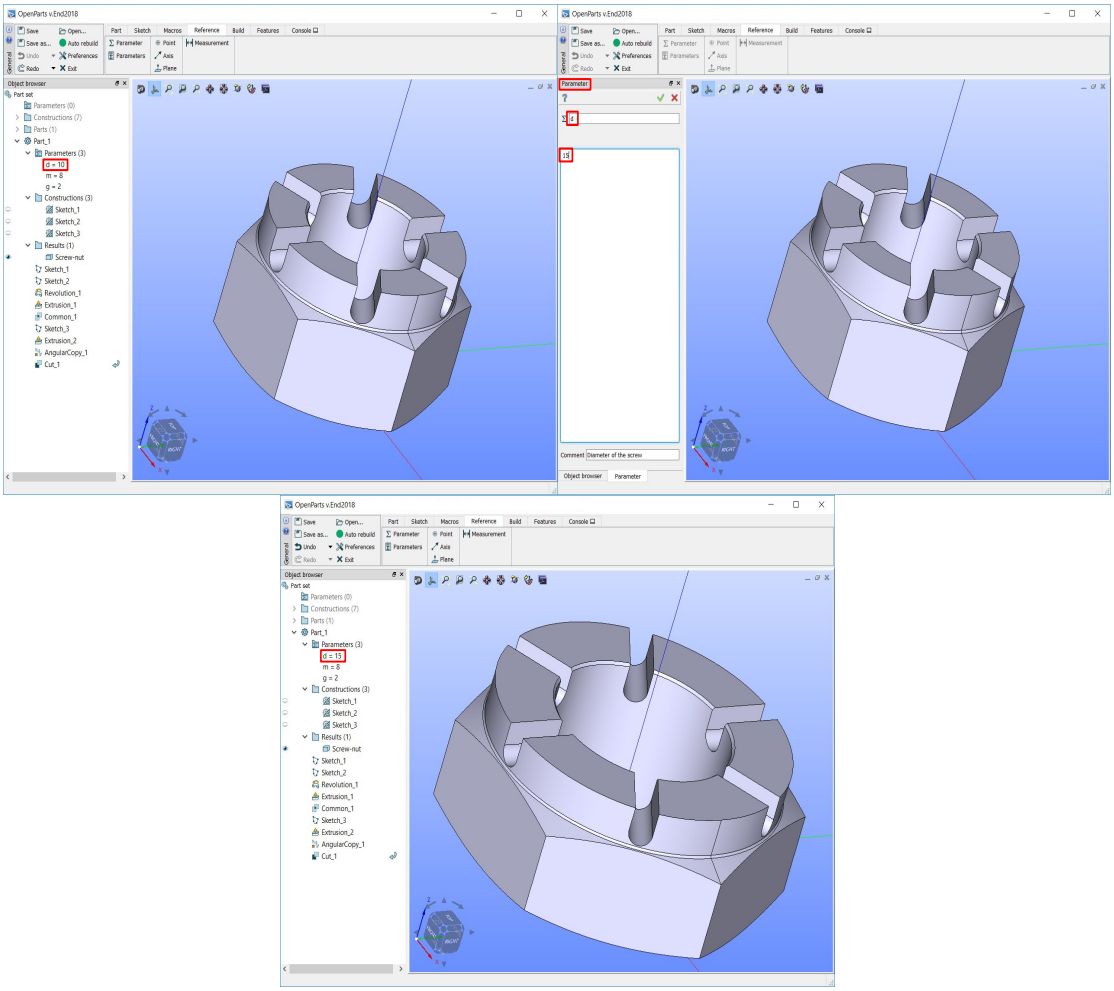


Figure 2.3: Example of parameter's update of the parametric CAD model of the screw nut shown in Fig. 2.2. The parameter that we want to edit (d) is selected in the picture on the top-left. The editing of the parameter is shown in the picture on the top-right. On the bottom, the CAD model updated with the new parameter's value is shown.

2. Implementation of the parametric CAD model by using a commercial CAD tool.

3. Utilisation of open source CAD libraries such as Open CASCADE[8].

The choice of developing a geometric kernel custimised to the particular test case has been done by several authors, such as Grasel et al. [22] and Numec [74]. Grasel et al. parametrise a turbomachinery blade with the parameters normally used by designers such as radii of trailing and leading edges, thickness distribution. Nemec [74] has implemented a software called *Cart3D* that allows to parametrise both 2D (airfoil) and 3D geometries such as fuselage, wing, canard and tail geom-

[8] `https://www.opencascade.com/`

etry. The CAD derivatives are computed with the FD method. Also Verstraete has implemented a CAD library called CADO [23] for parametrising turbomachinery components such as blades and cooling channels. Verstraete uses both gradient-based [24, 25] and gradient-free [23] optimisation methods. When utilising gradient-based algorithms, Verstraete computes the CAD derivatives in two ways: (i) by applying the AD technique to CAD kernel sources [24], (ii) by using the complex step method [25]. Vasilopoulos et al. [75] use an in-house CAD library called *Parablading* to parametrise a realistic compressor stator blade[9]. The CAD derivatives are computed with the FD method.

Haimes [76] defines a CAD-based framework by using the Computational Analysis Programming Interface (CAPRI). CAPRI provides a connection between CAD systems and CAE software by using an API that hides the vendor dependent CAD parameters. This API reads the features composing the CAD model and reconstructs the geometry with an internal CAD library. In this way, it is possible to parametrise a geometry in a commercial CAD system such as Pro/ENGINEER [77] and then exchange this geometry with CAE software. The main drawback of this approach is that the design technique provided by CAPRI to reconstruct the geometry is limited only to CAD loft. All other CAD features can be read but not changed. Fudge [77] employs CAPRI for optimising a DLR F6 wing-fuselage geometry.

Dannenhoffer and Haimes [26, 78–80] use the open source CAD-kernel Open CASCADE Technology (OCCT) as a geometric engine. They build a CAD model from Boolean operations of analytic CAD primitives, for which derivatives can be analytically calculated. The utilisation of this solution is straightforward for simple shapes (e.g. cylinders defined by origins, radii and axes) whilst is difficult to implement for complex ones. For these complex shapes, the derivatives are computed using FD.

Amstrong et al. [81] utilise as commercial CAD system CATIA. They use this commercial tool for parametrising geometries of different nature such as an air duct [20], a nozzle guiding vane [82], a transonic wing (ONERA M6) [82] and a car mirror [83]. They apply an incremental change to each parameter controlling the CAD model in order to generate a set of perturbed geometries. Every perturbed geometry serves to compute the finite difference approximation for the required CAD derivative. The main drawback of this choice is that the finite-size displacements could cause issues such as topological changes in the surface description and modifications in patch renumbering. In order to avoid these is-

[9] `http://aboutflow.sems.qmul.ac.uk/events/munich2016/benchmark/testcase3/`

sues, they project the geometry onto an STL approximation of the surface and the finite-differences of the displacements of grid nodes are evaluated on this STL. The utilisation of this approach is not straightforward in areas of high curvature and when considering components in close proximity. In these areas, a displaced node may be projected onto a part of the STL surface that corresponds to the wrong surface or location. This can occur because it is not straightforward to control the resolution of the STL.
Summarising, the scientific community has still to unanimously identify an approach to compute the shape derivatives which suits aerodynamic shape optimisation. Ideally, one would prefer to parametrise the geometry with commercial CAD systems, which are used by designers during the conception phase of the product. In this case, the shape derivatives have to be computed with the FD method. This method provides inexact derivatives calculation with a computational cost that scales with the number of design parameters. The development of a geometric kernel applied to a specific application field (such as the wing of an aircraft) allows the utilisation of several techniques for the calculation of the derivatives, such as complex variable method, exact differentiation and AD (Ch. 3). Its main limitation is that the extension of this geometric kernel to other application fields requires prohibitive development efforts. Also, if all the functionalities provided by CAD systems are not available, the imposition of manufacturing constraints such as the assembly ones is not possible.
In this thesis, another approach to compute the shape derivatives for parametric-based designs is investigated. The OCCT CAD kernel is used to build the parametric CAD models. This allows to impose all the manufacturing constraints. The CAD derivatives are computed by applying the AD technique to the sources of OCCT. The employment of the AD technique provides an exact calculation of the derivatives with the computational cost that is independent from the number of design parameters (if the reverse mode is implemented), as explained in Ch. 3.

2.3.2 NURBS-based parametrisation

NURBS-based parametrisations are CAD-based parametrisations which rely on NURBS [84]. NURBS definition is based on B-splines and Bezier representation.

A NURBS surface is defined as in Eq. 2.6:

$$\begin{aligned} R(u,v) =& \frac{N_{i,p}(u)N_{j,q}(v)W_{i,j}P_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)W_{i,j}} \\ & 0 \leq u \leq 1 \\ & 0 \leq v \leq 1 \end{aligned} \tag{2.3}$$

where $P_{i,j}$ represent the NURBS control points, $W_{i,j}$ are the *weights* and $N_{i,p}(u)$ and $N_{j,q}(v)$ are the B-spline basis functions of p-th and q-th degree, respectively. $N_{i,p}(u)$ and $N_{j,q}(v)$ are defined as in Eq. 2.4 and Eq. 2.5:

$$N^{i,0}(u) = \begin{cases} 1, & \text{if } u_i \leq u < u_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.4}$$

$$N^{i,k} = \frac{u - u_i}{u_{i+k} - u_i} N_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u) \tag{2.5}$$

The number of knots is $m = n + p + 1$. As shown in Eqs. 2.4 2.5, B-spline surfaces allow to control locally the surface since a change of a control point position/*weight* affects only the area $[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$.
By defining the piecewise rational basis functions $R_{i,p;j,q}(u,v)$ as in Eq. 2.6:

$$\begin{aligned} R_{i,p;j,q}(u,v) =& \frac{N_{i,p}(u)N_{j,q}(v)W_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)W_{i,j}} \\ & 0 \leq u \leq 1 \\ & 0 \leq v \leq 1 \end{aligned} \tag{2.6}$$

the NURBS surface $R(u,v)$ can be finally defined as in Eq. 2.7:

$$\begin{aligned} R(u,v) =& \sum_{i=0}^{m}\sum_{j=0}^{n} R_{i,p;j,q}(u,v)P_{i,j} \\ & 0 \leq u \leq 1 \\ & 0 \leq v \leq 1 \end{aligned} \tag{2.7}$$

Piegl [34] provides further details about NURBS. In aerodynamic shape optimisation, the control points and the weights of NURBS are used as design variables. An advantage of using NURBS for parametrisation is that NURBS allows a local control of the surface. As previously explained, a change in $P_{i,j}$ or in $W_{i,j}$ implies

a change only in the area defined by $[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$. Then, due to the continuity of the NURBS surface $S(u, v)$ in correspondence of the knots, $S(u, v)$ is $p - k$ times differentiable w.r.t. u at a u knot with multiplicity k. For the same reason, $S(u, v)$ is $q - k$ times differentiable w.r.t. v at a v knot with multiplicity k. Since NURBS utilise a very limited amount of CAD functionalities, NURBS-based design is vendor-neutral (i.e. the parametrisation can be exchanged from a CAD system to another one without loosing the design intents implemented by the designer).

A major drawback of using this approach is that the intuitive design parameters normally handled by the designer in the conception phase and defined within the CAD system (Subsec. 2.3.1) are not considered. Moreover, the imposition of the constraints such as the continuity between patches or the minimum thickness distribution is challenging even if possible [5, 85].

Several authors have used NURBS for shape optimisation. Martin [86, 87] defines a CAD-based shape framework where the design variables are only the control points of the NURBS (the weights are fixed) and the adjoint CFD solver is a continuous one (TAU) (see Ch. 6, Sec. 6.1). Bentamy and Guibault [88] use NURBS to parametrise an aircraft wing. They identify five sections such that every section is parametrised with a NURBS curve controlled by 24 control points. The final NURBS surface is obtained by interpolating the five NURBS curves. This parametrisation is used to aerodynamically optimise the wing. The optimisation results demonstrate that NURBS curves are able to provide the optimal shape with high aerodynamic performance while being controlled by a limited number of design parameters.

Zhang [5] also uses a NURBS-based approach (NURBS-based parametrisation with complex constraints, NSPCC) defined within an in-house CAD kernel to optimise several geometries such as a climate duct [89], a cooling channel for high-pressure turbine blades [89] and the ONERA M6 wing [11]. Unlike the previous methods, NSPCC allows the imposition of arbitrary constraints as follows. The NSPCC parametrisation imposes the patch continuity constraints (G0, G1..) between the NURBS. Then, the gradient of the resulting constraint matrix is calculated and the null space is extracted using the Singular Value Decomposition (SVD) factorisation. The singular vectors of this factorisation define the design parameters. Mykhaskiv utilises the NSPCC parametrisation approach and computes the shape derivatives with the differentiated open source CAD library Open CASCADE to optimise several geometries such as the side mirror of a car[10] and

[10] `http://aboutflow.sems.qmul.ac.uk/events/munich2016/benchmark/testcase4/`

the wing-body fairing of an aircraft [90].

2.4 Summary

This chapter has introduced the main parametrisation approaches used in shape optimisation. CAD-free parametrisations provide an easy way of computing the shape derivatives. The missing link to CAD software hinders the spread of these parametrisations in industry. The imposition of geometric constraints is not straightforward and the optimal shape is not provided in a CAD format such as STEP and IGES.
As opposed to CAD-free parametrisations, CAD-based parametrisations provide the optimised shape in a CAD format. This facilitates the reintroduction of the optimised shape in the industrial workflow. CAD-based parametrisations can be subdivided into two main families: parametric-based and NURBS-based designs. Parametric-based designs are defined by using parameters normally used by the designer to conceptualise the geometry whereas NURBS-based ones are defined starting from the generic description of the geometry delivered by exchange data format such as STEP and IGES. Parametric-based designs require important implementation efforts and are normally preferred for test cases which have been widely studied in literature. On the other hand, NURBS-based parametrisations provide a rich design space which is set up automatically. This type of parametrisation is rather preferred when investigating new designs.
The geometry of the test cases studied in this thesis is parametrised with the parametric-based design approach. This design approach provides the main benefit which is to encode the design intents. This is achieved by controlling the geometry with the engineering parameters normally handled by the designer during the conception phase of the product. Also, parametric-based design allows to impose the manufacturing constraints by using the features provided by the CAD system. This is important to easily insert the optimised shape into the design workflow.

Chapter 3

Computation of the shape derivatives

This Chapter presents the approach used in this research work to calculate the shape derivatives. Sec. 3.1 presents the non-AD approaches. Sec. 3.2 introduces the AD technique. Sec. 3.3 describes **Open CASCADE Technology (OCCT)**, the CAD kernel utilised in this thesis. Sec. 3.4 shows the application of the AD tool ADOL-C to the sources of **OCCT**. The reverse mode version of **OCCT** will be used throughout this research to compute the shape derivatives.

3.1 Derivatives: non-AD methods

This section presents the non-AD methods which can be used to calculate the derivatives. These methods are: (i) finite difference (FD) method, (ii) complex variable method and (iii) exact differentiation.
Finite Difference method provides the calculation of the derivatives as in Eq. 3.1:

$$\frac{\partial J}{\partial x} \approx \frac{f(x+\delta x) - f(x)}{\delta x} \tag{3.1}$$

The main advantage of FD method is that it can be easily used to compute the derivatives with black-box commercial solvers. On the other hand, the choice of δx is not straightforward. A too small δx implies an high round-off error. On the other hand, a too big δx causes a large truncation-error. Moreover, this method requires the calculation of the objective function f in order to compute the derivative of f w.r.t. a single input variable. As a consequence, if the number of input variables is n, the function f has to be computed n-times to provide the computation of all the derivatives. In other words, by using FD, the computational cost scales with

the number of input variables.
The complex variable method [91] provides the computation of the derivatives by using an analytic expansion, Eq. 3.2:

$$f(x+\epsilon i) = f(x) + \epsilon\frac{\partial f}{\partial x}i - \frac{\epsilon^2}{2!}\frac{\partial^2 f}{\partial x^2} - \frac{\epsilon^3}{3!}\frac{\partial^3 f}{\partial x^3}i + \frac{\epsilon^4}{4!}\frac{\partial^4 f}{\partial x^4}\cdots \tag{3.2}$$

The imaginary parts are finally equalised to calculate the derivative of f w.r.t. x, Eq. 3.3:

$$\begin{aligned} Im[f(x+\epsilon i)] &= \epsilon\frac{\partial f}{\partial x} - \frac{\epsilon^3}{3!}\frac{\partial^3 f}{\partial x^3}i + \mathcal{O}(\epsilon^5) \\ \frac{\partial J}{\partial x} &= \frac{Im[f(x+\epsilon i)]}{\epsilon} + \mathcal{O}(\epsilon^2) \end{aligned} \tag{3.3}$$

such that, w.r.t. FD, the subtraction between two quantities which are almost equal (i.e. very small values of ϵ) is not considered. This allows to avoid the round-off error w.r.t. FD method. If the step size is carefully chosen, the complex variable method provides the value of the derivative very close to the machine precision. The computational costs of C-V still scales with the number of design variables and is further increased by the complex variable arithmetic.
Exact differentiation method computes the derivatives analytically, i.e. by hand. This approach, which allows to compute the exact value of the derivatives, needs the knowledge of the exact equations of the model. Moreover, it would require huge efforts spent in manual programming, which is tedious and error-prone.
Summarising, all the methods mentioned above (finite difference, complex variable and exact differentiation) have a computational cost which scales with the number of design variables. Moreover, not all of these methods provide an exact calculation of the derivatives. If the sources of the program are available, the reverse mode implementation of AD allows to exactly compute the derivatives with the computational cost which is independent from the number of design variables.

3.2 Computation of the derivatives with AD

This section presents AD[1], the technique used in this research to compute the shape derivatives. Subsec. 3.2.1 introduces the fundamental principles of AD. Subsec. 3.2.2 explains the possible ways AD can be implemented and Subsec. 3.2.3 describes the AD tool used in this thesis, ADOL-C [92].

[1] `http://www.autodiff.org/`

3.2.1 AD fundamental principles

Given a code C implementing a differentiable function $F : X \in \mathbb{R}^n \rightarrow Y \in \mathbb{R}^m$, AD is a set of techniques that, applied to C, provides a new code C'. This new code C' calculates both the function F and the derivatives of F w.r.t. the input variables, F'.

The fundamental principle of this technique relies on the fact that the code C can be viewed as a sequence of elementary functions $\{f_1; \dots f_p;\}$ (i.e. simple operations such as addition, multiplication..). Therefore, the function F can be expressed as in Eq. 3.4:

$$Y = F(X) = f_p(\dots(f_2(f_1(X)))\dots) \tag{3.4}$$

F' can be exactly calculated by computing the derivative of every function f_i w.r.t. the input X and concatenating these derivatives using the chain rule, Eq. 3.5:

$$\begin{aligned} F'(X) =& (f'_p \circ f'_{p-1} \circ f'_{p-2} \circ \dots \circ f'_1(X)) \\ & .(f'_{p-1} \circ f'_{p-2} \circ \dots \circ f'_1(X)) \\ & \dots . \\ & .(f'_1(X))) \end{aligned} \tag{3.5}$$

AD can be applied in two modes: forward and reverse mode.

Both forward and reverse mode calculate a directional derivative. The forward mode computes each column of the Jacobian at once by multiplying the Jacobian ∇F with the directional (or weighting) vector $\dot{X}$, Eq. 3.6 [93]:

$$\nabla F \cdot \dot{X} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_p}{\partial x_1} & \frac{\partial f_p}{\partial x_2} & \dots & \frac{\partial f_p}{\partial x_n} \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} \tag{3.6}$$

Given the n inputs x_j $(j = 1 \dots n)$ to F, an invocation of the differentiated chain allows to compute a column of the Jacobian such that n invocations are needed to compute all the derivatives. Normally, in aerodynamic shape optimisation problems n (i.e. number of input variables) is higher then m (number of cost functions). This makes the forward mode inefficient for typical aerodynamic shape optimisation applications.

The reverse mode transposes the Eq. 3.6 in order to compute the vector-matrix

product shown in Eq. 3.7:

$$\bar{Y}\nabla F = \left[\bar{f}_1, \bar{f}_2, \dots \bar{f}_p\right] \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_p}{\partial x_1} & \frac{\partial f_p}{\partial x_2} & \cdots & \frac{\partial f_p}{\partial x_n} \end{bmatrix} = [\bar{x}_1, \bar{x}_2, \dots \bar{x}_n] \tag{3.7}$$

Per every invocation of $\bar{f}$, the reverse mode provides one row of the Jacobian, i.e. the derivative of the cost function w.r.t. all the input variables. This makes the reverse mode more efficient then the forward mode to calculate the derivatives for typical aerodynamic shape optimisation problems.

3.2.2 AD implementations

AD can be applied by using two main options: Source Transformation (S-T) and Operator-Overloading (O-O). The first option interprets (i.e. parses) the statements of the original source code and then produces modified source code that adds the statements necessary to compute the derivatives. Most popular S-T codes are Tapenade [94], TAF [95] and Open AD/F [96]. This differentiation option is normally applied to programming languages such as C and fortran.
S-T is not implemented for programming languages such as C++ because it is too complex. For such languages, O-O is utilised. O-O allows to implement extensions of standard operations (e.g. +, -, *...) for previously defined data-types. Code listing 3.1 shows the definition in C of a new data type (`adouble`) that stores both a value and the derivative information. An example of overloaded multiplication

```
struct {
  double value;
  double ADvalue;
} adouble
```

Code Listing 3.1: `adouble` definition.

in C++ between two `adouble` is shown in code listing 3.2: where `value` stores the value of the variable and `ADvalue` stores the derivative information.
All "intermediate variables" (i.e. all `real` variables that depend on the input parameters and influence the cost functions) have to be defined with the extended data-type. The derivative information, therefore, can be computed along with the calculation of the function to be differentiated. This suits forward mode differentiation, where the derivatives are computed together with the evaluation of the function. On the other hand, for the calculation of the derivatives in the reverse

```
adouble operator * (adouble a, adouble b)
{
  adouble prod;
  prod.ADvalue = a.value*b.ADvalue + a.ADvalue*b.value;
  prod.value = a.value*b.value;
  return prod;
}
```

Code Listing 3.2: Example (simplified) of multiplication between two `adouble`.

mode, O-O software perform a first sweep of the code to tape all the operands and operations utilised in the code. The output variable seeds are defined at the start of this tape. Then, the appropriate reverse differentiated statements are invoked for each primal operation stored in the tape such that, finally, the reverse mode operation is carried out with the taped primal and the reverse-propagated derivative variables.
The CAD library (**OCCT**) that we want to differentiate in this research work is an object-oriented C++ library. Therefore, it is differentiated by using O-O, which allows to keep the main benefits of object-oriented architectures such as abstraction and code recycling.

3.2.3 ADOL-C

The main AD tools for C++ are: (i) ADOL-C [92], (ii) codipack [97], (iii) FADBAD [98] and (iv) CppAD [99]. ADOL-C is the one chosen in this research. This O-O software is an open source library most widely used and most mature for C/C++. It has been used for a wide variety of applications such as bevel gear simulations [100], design of non linear controllers [101], optimal control problems [102] and shape optimisation [103].
ADOL-C features the following options and differentiation modes:

- *traceless* (differentiation mode: forward).
- *trace mode* (differentiation modes: forward or reverse).

The data type defined within ADOL-C to store both a value and its derivative (code listing 3.1) is `adouble`.

Traceless mode

The *traceless mode* performs the computation of the derivatives of a function F w.r.t. an input X along with the calculation of the "primal" (i.e. the calculation

of the function F). Both the value and its derivative are stored in the same data type (i.e. `adouble`, code listing 3.1). Considering that no tape of operations and operands is defined, the only way to compute the derivative with the *traceless* approach is the *forward mode*.
Given the function in Eq. 3.8:

$$f(x_1, x_2) = sin(x_1) + x_1 \cdot e^{x_2} \tag{3.8}$$

code listing 3.3 presents the program that implements the function in Eq. 3.8 whereas code listing 3.4 shows the differentiated version of this software (the "differentiated program").

```
double f(vector<double> inputParams)
{
  double *y = new double(); // output
  double *w = new double[6]; // intermediate variables
  w[0] = inputParams[0]; w[1] = inputParams[1];
  w[2] = exp(w[1]); w[3] = w[0] * w[2];
  w[4] = sin(w[0]);
  w[5] = w[3] + w[4];
  y = w[5];
  return y; // return the value of the function
};
```

Code Listing 3.3: Implementation of the function f in Eq. 3.8.

The differentiated program includes the header file that implements the data type `adouble` for the *traceless forward mode* (`adtl.h`) and defines the input parameters, the intermediate variables and the cost function with this data type `adouble`. The design parameter w.r.t. the derivative has to be computed (independent variable `x[0]`) is set by using the `setADvalue(1)` method and the derivative value of the cost function ($derValue$) is extracted by calling the method `getADvalue()`.

```
#include <adtl.h>

typedef adtl::adouble adouble;

adouble f(vector<adouble> inputParams)
{
  adouble *y = new adouble(); // output

  adouble *w = new adouble[6]; // intermediates

  w[0] = inputParams[0]; w[1] = inputParams[1];
  w[2] = exp(w[1]); w[3] = w[0] w[2];
  w[4] = sin(w[0]);
  w[5] = w[3] + w[4];
  y = w[5];

  return y;

};

// set the first variable inputParams[0] as independent variable.
inputParams[0].setADvalue(1);
//extract the value of "f"
adouble ADvalue = f(inputParams);
// extract the derivative of "f" w.r.t. inputParams[0]
double derValue = ADvalue.getADvalue();
//unseed inputParams[0]
inputParams[0].setADvalue(0);
```

Code Listing 3.4: Forward mode differentiation of the function f implemented in the code listing 3.3.

Trace mode

The computation of the derivatives by using the *reverse mode* requires the creation a tape of operations and operands. After the first sweep that allows to store this tape, a second sweep is run backwards (from output to input variables) to calculate the derivatives.
ADOL-C allows the definition of the tape by performing the following steps. First, the header-file (`adolc.h`) that implements the new data type `adouble` is included. Second, the part of the code that has to be differentiated (i.e. that is required for the computation of the derivatives) is taped by using the instructions "`trace_on(1);`" and "`trace_off();`".

Code listing 3.5 provides the differentiation in trace mode of the function f w.r.t. the input variables `inputParams`. In this code, the variable `yp` defines the objective function to be differentiated.
After having traced the code necessary for the computation of the derivatives (i.e. all the operations and the operands have been stored in a tape), it is possible to proceed with the calculation of the derivatives. First, the independent variables `x` are defined and then the drivers `jac_vec(...)` (which calculates the product

```
#include "adolc.h"

//store the operations in a tape

int trace_tag = 1;
void f_tape()
{

double *yp = new double(); // output

trace_on(trace_tag); //start taping

//initiate input parameters
inputParams[0]<<=inputParams[0].getValue();
inputParams[1]<<=inputParams[1].getValue();

adouble ADvalue = f(inputParams);

ADvalue >>= yp;

trace_off(); //stop taping

}

vector<double> der_fwr(2), rev_der(2);
f_tape()

double *x = new double(); //independents

x[0] = inputParams[0].getValue();
x[1] = inputParams[1].getValue();

double* tangentVec = [1, 0]; // size tangentVec = # input parameters
double* rangeVector = [1]; // size rangeVector = # cost functions

// Forward/reverse calculation of the derivatives
// yp.size() = 1, x.size() = 2
jac_vec(trace_tag, yp.size(), x.size(), x, tangentVec, der_fwr);
vec_jac(trace_tag, yp.size(), x.size(), x, rangeVec, rev_der);

//utilise the gradients stored in der_fwr/rev_der
```

Code Listing 3.5: Reverse mode differentiation of the function in the code listing 3.3

.

Jacobian × vector, forward mode) and `vec_jac(...)` (which calculates the product vector × Jacobian, reverse mode) are executed. `jac_vec(...)`/`vec_jac(...)` allows to compute the columns/rows of the Jacobian such that, to calculate the full Jacobian matrix, `jac_vec(...)` has to be executed n times and `vec_jac(...)` is run m times.
In aerodynamic shape optimisation, the number of cost functions m is much lower then the number of design variables n. The reverse mode driver (`vec_jac(...)`) is therefore used to perform the computation of the derivatives.

3.3 CAD kernel used in this research: OCCT

OCCT[2] is the only open source CAD library available on the market. This makes this library suitable to the application of the AD technique. It is a "software development kit" intended for development of applications dealing with 3D CAD data or requiring industrial 3D capabilities, and designed for rapid production of sophisticated domain-specific software dealing with 3D models in design (computer-aided design, CAD), manufacturing (computer-aided manufacturing, CAM), or numerical simulation (computer-aided engineering, CAE).
OCCT is an object-oriented C++ class library. The C++ classes can be divided into:

- basic data structures (geometric modeling, visualisation, interactive selection and application specific services),
- modeling algorithms,
- working with mesh (faceted) data,
- data interoperability with neutral formats (IGES, STEP).

The C++ classes are organised in packages which are grouped in toolkits (libraries). Finally, the toolkits compose the seven modules of the **OCCT** modular structure (Fig. 3.1). The Foundation Classes module is the base of all other **OCCT** classes. It contains primitive types such as `Standard_Boolean` (the data-type to define boolean operations), `Standard_Real` (data-type for `double`), `Handle` (the smart pointer implemeted in **OCCT**), `TCollection` classes (which group statically or dynamically sized aggregates of data such as arrays, lists etc.). This module also provides numerical algorithms and basic algebra calculations

[2] `https://www.opencascade.com/`

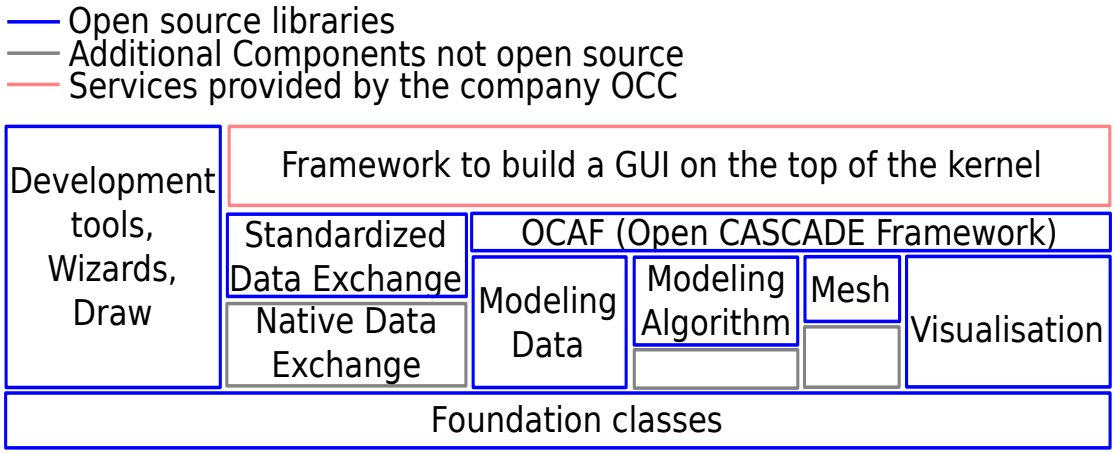


Figure 3.1: Simplified diagram of **OCCT** kernel organisation from Open CASCADE website.

such as transposition of vectors and matrices, solving linear systems etc.
The Modeling Data module provides the data structures for the 2D and 3D geometric primitives such as `Geom_Curve` and `Geom_Surface`. This module also implements the "boundary representation" (BRep) of the 3D objects. BRep encodes the shape as an aggregation of geometry within topology.
Another important package is `TopExp`, which allows to explore all the topological entities stored in a topological shape. For example, one can implement a function which takes as input a certain geometry and provides as output the list of points/curves/surfaces that compose the initial geometry. This is useful to analyse exchange files such as IGES and STEP which store the geometrical and the topological information of a CAD model.
The Modeling Algorithms module consists of the geometrical and topological algorithms provided by **OCCT**. This module presents low-level geometric tools for computing intersection between curves/surfaces (`GeomAPI_IntCS`), projection of points/curves on curve/surface (`GeomAPI_ProjectPointOnCurve/Surface`), construct free-form curves and surfaces based on constraints (e.g. `AppDef_Variational`). This module also implements algorithms to tessellate shapes, determinate the properties of shapes (e.g. barycentric point of a face `BRepGProp`), convert shapes to NURBS (`BRepBuilderAPI_NurbsConvert`) and sew adjacent shapes. Finally, top-level Application Programming Interfaces (API) for the immediate construction of primitives such as prisms, cylinders and cones (`BRepPrimAPI_MakePrism`, `BRepPrimAPI_MakeCone`) are also provided. Several of these classes are used to implement the CAD models of the geometries of the two test cases (code listings B.1 B.2).
The Mesh module offers the possibility to compute a surface mesh. The Visuali-

sation module supplies the mechanisms necessary to visualise the data. The Data Exchange module allows to exchange (i.e. read/write) shapes by using popular data formats such as STEP and IGES. The Application Framework module provides the possibility to manage application-specific data (user attributes) and commonly used functionality (save/restore, undo/redo etc.). The number of **OCCT** developers is around 100 and the number of downloads per year is usually between 6000 and 7000 [104]. Several companies and centres of research adopt OCC solutions. The main clients are Électricité de France (EDF), Framatome and Commissariat à l'Energie Atomique et aux Energies Alternatives (CEA). A complete list of OCC customers is presented on OCC website[3].
The features provided by **OCCT** justify the utilisation of this library as modeling kernel for the implementation of the parametrisations used in this thesis. Sec. 3.4 explains how the AD tool ADOL-C has been applied to **OCCT** sources.

3.4 Automatic Differentiation of OCCT

This section explains how the AD tool ADOL-C (Subsec. 3.2.3) has been applied to **OCCT**. Mladen Banovic has been the responsible for the development of this part of the research.
Several possible ways of source code modification have been considered for ADOL-C integration [105] in **OCCT**, but one has been taken as a way to proceed with the full sources - the *typedef* approach. The *typedef* is a specifier provided by C++ which allows to create an alias of a type name. This alias can substitute the original type name throughout the program.
The idea of the *typedef* approach is to replace all `doubles` by `adoubles` (where the `adouble` is the data type defined in ADOL-C), by using a *typedef* already existent in **OCCT**, the `Standard_Real`. The main difference with the differentiation approach used in code listings 3.4 3.5 (where only the variables involved in the calculation of the derivatives are defined using the data type `adouble`) is that with the *typedef* approach all the `double` variables are defined as `adouble`. The principal advantage of the *typedef* approach is that code modification should be minimal. The drawback is that also the variables that do not need to be differentiated (i.e. are not part of the computational chain) will be extended to the new type `adouble`. This can increase the memory requirements of the code.
Although the idea about the *typedef* approach looks simple, it is not as straight-

[3] `https://www.opencascade.com/content/trusted-industries`

forward as one would expect. The utilisation of the data type `adouble` implies a significant amount of code modification due to compile-time errors and run-time ones. Some of the compile-time issues are related to: the mismatch between `adouble` size and the size of $2 \times Standard\_Integer$, the impossible conversion from `adouble` to `int` data type, the impossibility to use `adouble` with functions that are part of external libraries or with functions for printing an output such as *sprintf*. The run-time issues are related to several types of issue such as the usage in **OCCT** of the C dynamic memory allocation that causes errors once the `adoubles` are present.
After integrating the *traceless* forward mode into **OCCT** by using the *typedef* approach, the reverse mode of AD has also been completed. The first step is to compile **OCCT** sources with the ADOL-C *trace* headers, which has been completed successfully. The following step is to use ADOL-C driver functions in the specific parts of the code in order to evaluate the derivatives. Further details about the differentiation of the **OCCT** CAD kernel are presented by Banovic et al. [105]. Sec. 5.6 (U-bend test case) and Sec. 5.9 (compressor stator test case) present the validation of the derivatives and the analysis of both the run-time ratios and the memory requirements for the two CAD models implemented for this research.

3.5 Summary

This Chapter has focused on the differentiation of the CAD kernel Open CASCADE Technology. Firstly, an introduction to the AD technique and to ADOL-C (the AD tool utilised in this research) has been provided. Secondly, the open source CAD library OCCT has been described. This library provides several modules which implement geometrical and topological algorithms for projecting points onto curves/surfaces, constructing free-form surfaces, converting shapes to NURBS. OCCT also implements surface mesh generators, data exchange classes used to read/write shapes with data format such as STEP and IGES. All these modules make OCCT a complete CAD kernel whose functionalities can be used to parametrise any typical industrial shape.
The approach utilised to differentiate OCCT has been based on the extension of all the `real` variables to the data type `adouble`, which is defined within the differentiation tool ADOL-C and stores both a value and its derivative information. OCCT has been differentiated in both traceless forward mode and trace reverse mode. The reverse mode is computationally more convenient if the number of input variables (i.e. design parameters) is much higher then the number of output

variables (i.e. cost functions). This is the case of aerodynamic shape optimisation problems. The differentiated version of OCCT in reverse mode will be therefore used in this research work.

Chapter 4

Re-parametrisation tool

Aerodynamic shape optimisation is usually applied to CAD geometries that are normally not equipped with a shape parametrisation, or the parametrisation is arising from a particular design workflow and does not offer design variables that are suitable for optimisation. In some cases, the CAD geometry is only available in data exchange format such as STEP files. The STEP file does not store all the CAD features utilised to implement the parametric CAD model (Subsec. 2.3.1). Therefore, the first step is to develop the parametric CAD model and to then fit the parameters implemented within this CAD model to best approximate the baseline shape. The derivatives provided by the differentiated **OCCT** can be utilised to solve this fitting problem, as demonstrated in the following sections. Sec. 4.1 explains the main approaches which can be used to parametrise an airfoil. Sec. 4.2 presents the parametrisation chosen in this research ("the proposed parametrisation") and Sec. 4.3 provides the optimisation problem solved to fit the "proposed parametrisation" to the airfoil profile given by the TUB test case (Sec. 5.7).

4.1 An investigation about airfoil parametrisations

In literature, several approaches are proposed to parametrise an airfoil [106–109]. Among these works, Castonguay et al. [107] compares four possible parametrisations to be used in gradient-based optimisations. These parametrisations are: (i) node-based, (ii) B-spline curves, (iii) Hicks-Henne bump functions, (iv) PARSEC method.

Node-based parametrisations have already been introduced in Sec. 2.2. They use as design variables the 2D mesh point coordinates sampled onto the airfoil profile. This allows the definition of a very rich design space. Two are the main drawbacks of this approach: (i) the optimised profile could be characterised by

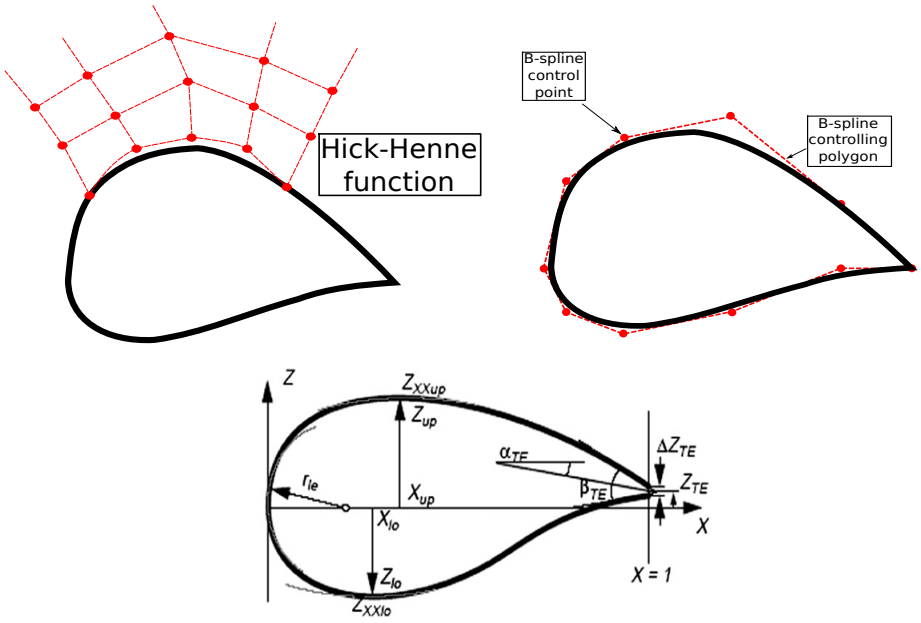


Figure 4.1: Approaches to parametrise an airfoil: Hick-Henne approach (top-left), B-spline approach (top-right) and PARSEC approach [110] (bottom).

unwanted unsmoothness such that the regularisation of the profile is required, (ii) no designer-friendly parameters are defined.

The Hicks-Henne functions allow to parametrise the 2D profile with a set of weighted Hicks and Henne sine "bump" functions, as shown in Fig. 4.1. This results in a design space which considers a reduced number of parameters w.r.t. the node-based approach. Moreover, the computed gradients always remain smooth. As reported by Castonguay [107], while solving a typical inverse problem such as the one of finding the 2D profile that generates a fixed pressure distribution target, the Hicks and Henne functions are not always able to match the initial 2D profile. This demonstrates that this approach inhibits the section from assuming a broad variety of shapes within which the shape with best performance can be identified. Moreover, the presence of intuitive design parameters is still missing.

The B-spline approach (top-right in Fig. 4.1) allows to describe the 2D profile with a/a set of B-spline curve/s. The B-spline curves [34] are typical features of CAD systems and have several advantages compared to the previous two approaches described above. Firstly, the degree of the polynomial is limited at a user-defined value. Secondly, the parametrisation is controlled by a limited set of parameters. Thirdly, every control point of the B-spline governs a particular region of the profile such that a local change in the curve can be done without affecting the other parts of the profile and while assuring the smoothness of the geometry. Unfortunately, despite all these advantages, this way of applying the

B-spline curves does not provide parameters that the designer has in mind during the conception phase such as thickness distribution and camberline. The author will highlight in Sec. 4.2 that, if properly arranged, the B-spline curves could be used to define user-friendly parameters.
The PARSEC method [111] (bottom in Fig. 4.1) considers as design variables a set of controlling parameters that the designer has in mind during the conception phase. These parameters are: the leading edge radius (r_{LE}), upper crest location (X_{UP}, Z_{UP}), lower crest location (X_{LO}, Z_{LO}), upper and lower curvature (Z_{xxUP}, Z_{xxLO}), trailing edge coordinate(Z_{TE}) and direction(α_{TE}), trailing edge wedge angle(β_{TE}) and thickness(ΔZ_{TE}). Despite the mentioned parameters being conceptually close to the requirements of the designer, this method provides a design space which consists just of 11 parameters while a proper design space should be characterised by 20/25 parameters [108]. Moreover, Castonguay demonstrates that the 2D section proposed by PARSEC is not able to change the leading edge shape of a predefined profile (ONERA M6) in order to match a target pressure distribution.

Despite its limitations, the PARSEC method identifies some of the main requirements done by the designer when implementing an airfoil parametrisation. Ideally, a designer would prefer to work with a parametrisation that consists of a limited set of "physical design parameters" (as the radius at the leading edge of the airfoil) which allows to pick up the main shape modes of the geometry.
The airfoil parametrisation adopted in industry [75] (shown in Fig. 4.2) consists of: a 2D line, the *camber-line*, that usually is a B-spline curve, the thickness distribution of the suction and the pressure side along the axial chord (these two distributions are usually B-Spline curves and can coincide in case the designer wants to control the suction and the pressure side by using a single law) and two arcs of circle to design the leading and trailing edge area.
The design parameters of this type of parametrisation are: the camber-line control points or directly its angle distribution β (the camber-line angle defines the geometrical inlet/outlet angle of the blade), the control points that control the thickness distribution of the suction and the pressure side and the radii of the trailing and leading edge arcs.
The main drawback of this parametrisation approach is that it does not guarantee the G2 continuity of the 2D profile along all the 2D section. Given two curves C_1 and C_2 that are sharing a point P, C_1 and C_2 are *G2 continuous* in P if $k_{C_1}(P) = k_{C_2}(P)$, where k_{C_1} is the curvature of C_1 and k_{C_2} represents the curvature of C_2 [112]. For the parametrisation shown in Fig. 4.2, the G2 conti-

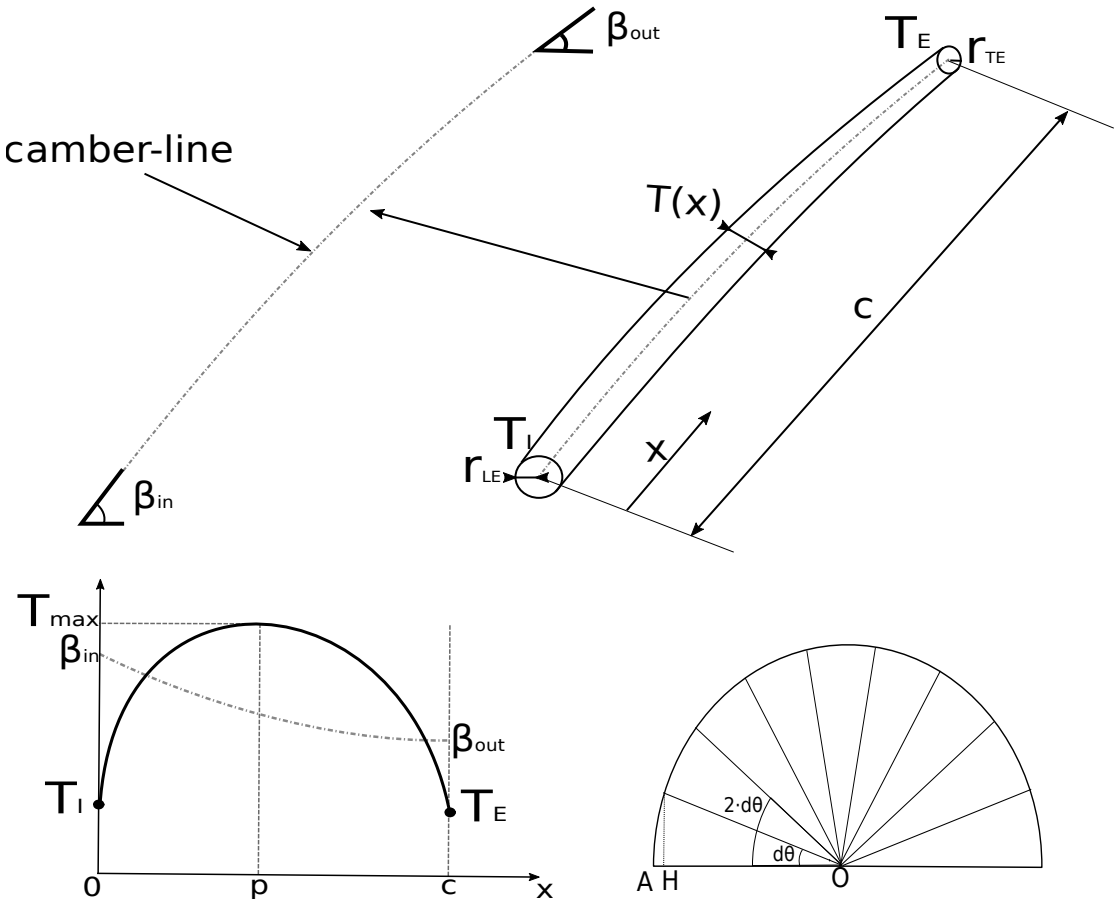


Figure 4.2: Top: camber-line of a typical airfoil parametrisation approach adopted in industry and the typical airfoil parametrisation implemented by designers. Bottom Left; typical airfoil parametrisation thickness distribution. Bottom right: graphical Representation of the Law (AH value) used to distribute the points on the camber-line for the definition of the suction and pressure Side of the 2D Section.

nuity is not imposed at the points of connection between the LE/TE arc and the suction/pressure side. During the flow optimisation, the optimiser could provide a thickness distribution that allows to have the G2 continuity at these connection points. This would imply that both suction and pressure sides would have a shape close to two circular arcs (this to respect the curvature continuity with the leading/trailing edge arcs). Finally, the optimised 2D section would consist of four arcs, reducing the possibilities to identify the shape with best performance.
The parametrisation used by the author in this thesis, which guarantees by construction the G2 continuity along all the 2D profile, is explained in Sec. 4.2.

4.2 2D Profile Parametrisation

The parametrisation proposed in this research utilises only B-Spline curves to define the typical industrial parameters such as the leading and trailing edge radius. The 2D section of the blade is generated using a camber-line (shown in Fig. 4.3) represented by a B-spline curve and characterised by seven control points. Eight reference points $(P1, ..., P8)$ are distributed on it. The distribution of these eight points can be either uniform or based on a stretching function. In this case, the author opted for the second one, choosing in particular a cosine distribution (Fig. 4.2):

$$AH = -AO + AO \cdot cos(i \cdot d\theta) \tag{4.1}$$

$$d\theta = \frac{\pi}{N+1} \tag{4.2}$$

where $i = 1 \dots N_p$ and N_p is the number of points to be distributed on the camber-line (in this case 8). Based on the several AH values, the points $(P1, ..., P8)$ are positioned on the camber-line as shown in Fig. 4.3.
This choice is done to cluster points near the leading and trailing edge (LE and TE) of the camber-line. The control points for the suction and pressure B-splines curves are generated as equidistant offsets of the reference points perpendicular to the camber-line (Fig. 4.3). Finally, the suction and pressure curves are smoothly joined (*continuity G2*) using the specified radius of curvature at the trailing and the leading edge by utilising the approach explained above. The imposition of the *G2 continuity* at the LE and TE is possible based on a particular property of the B-spline curves.

The formulation of the B-spline curve is expressed in Eq. 4.3:

$$\begin{gathered} C(u) = \sum_{i=0}^{l} P_i N_{i,p}(u) \\ 0 \leq u \leq 1 \end{gathered} \tag{4.3}$$

where $C(u)$ represents the coordinate of geometry vector, p is the order of the curve, P_i is the B-spline control points vector, l are the number of control points and $N_{i,p}(u)$ is the B-spline basis function [113] defined on the non-decreasing knot

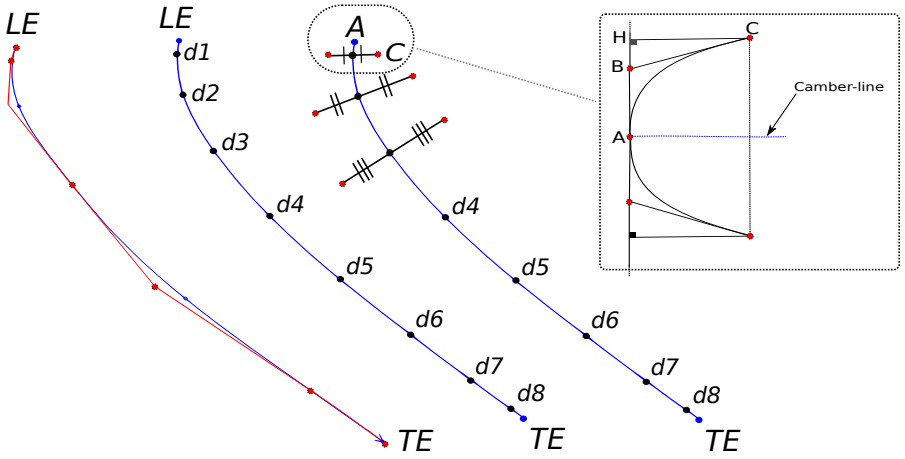


Figure 4.3: Left: Camber-line (blue) with corresponding control polygon (red) and uniform point distribution; Right: Construction of pressure/suction control points; Imposition of curvature (*G2 continuity*) at the LE

vector U (Eq. 4.4):

$$U = [u_0, \ldots, u_r] \, \epsilon \, \mathbb{R}^{r+1} \quad r = l + p + 1 \tag{4.4}$$

The AB length [34] of the B-spline curve shown in Fig. 4.3 can be computed as in Eq. 4.5:

$$AB = \sqrt{\kappa_A \cdot \frac{u_{p+1}}{u_{p+2}} \cdot \frac{p-1}{p} \cdot CH} \tag{4.5}$$

where κ_A is the curvature of the point A in the 2D geometric space where the airfoil is defined, AB represents the distance between control point A and B and CH is the distance of control point C from the AB line. Therefore, it is possible to impose the curvature in point A by using the formula in Eq. 4.6:

$$\kappa_A = \frac{AB^2}{CH} \cdot \frac{u_{p+2}}{u_{p+1}} \cdot \frac{p}{p-1} \tag{4.6}$$

This approach is applied to suction and pressure B-splines. In particular, the two curves have the same radius of curvature at the LE (the point A in Fig. 4.3). This radius is controlled as design parameter of the optimisation. The same approach is also used for the TE radius such that the *G2 continuity* is kept along all the section.

In summary, the 2D profile parametrisation consists of 23 parameters of which, 10 parameters control thickness (2 of them are the radii of TE and LE) and 13 parameters control the camber-line movement, as shown in Fig. 4.4. This parametrisation provides a design space rich enough to perform the aerodynamic

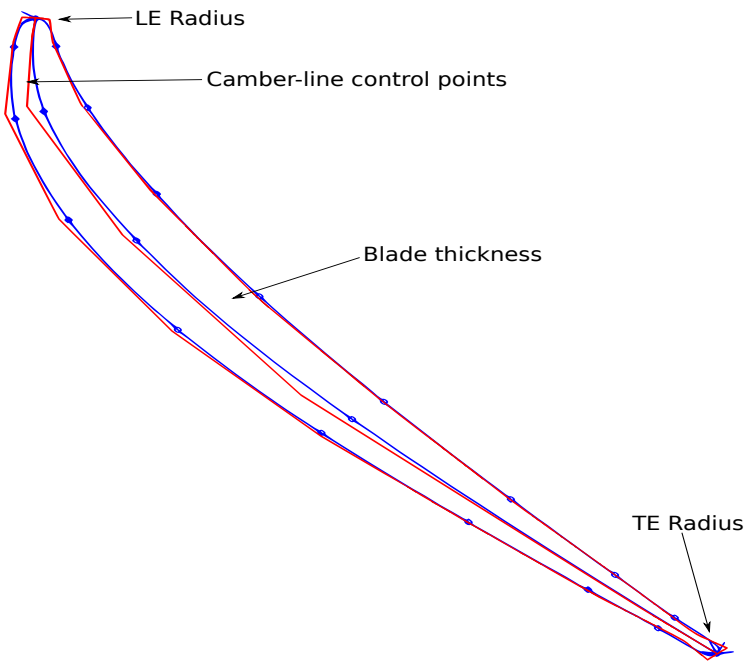


Figure 4.4: Parameters controlling the airfoil.

shape optimisation. If we would consider a number of design parameters reduced w.r.t. the 23 parameters utilised in this research, the parametrisation would be very similar to the one provided by the PARSEC method. The design space provided by the PARSEC method does not suit shape optimisation, as reported in Sec. 4.1. On the other hand, the increase of the number of design parameters is not necessary. As highlighted by Masters et al. [108], the design space very similar to the parametrisation proposed in this thesis can be considered rich enough to perform the aerodynamic shape optimisation of an airfoil. In this research, this is verified by comparing the optimal shape provided by the parametric-based design (i.e. the 2D profile parametrised with 23 design parameters) with the one given by a 2D section controlled by an enlarged design space (41 design parameters, Sec. 7.7).

4.3 2D fitting problem

This section provides the algorithm developed by the author to refit the existent 2D profile of one of a test cases investigated in this thesis, the Technical University of Berlin (TUB) Stator blade[1] (Sec. 5.7). This 2D profile is defined as the *target section* while the parametrisation explained in Sec. 4.2 (Fig. 4.4) is the *proposed*

[1] `http://aboutflow.sems.qmul.ac.uk/events/munich2016/benchmark/testcase3/`

parametrisation.
The *target section* consists of two curves (i.e. suction and pressure sides). These two curves have been extracted out of the CAD model which stores the geometrical/topological information of the blade (Sec. 5.8).
The fitting problem can be divided in the following steps:

- sample a set of points nb_pnts (in this case $nb\_pnts = 1500$ [114]) onto the first curve (e.g. pressure side) of the *target section*.
- Store all these points P_i, with $i = 1...nb\_pnts$.
- Project the set of points P_i onto the pressure side of the *proposed parametrisation*and store the set of projected points T_i.
- Define the cost function as the least square distance $f(x)$ (Eq. 4.7).
- Repeat the same procedure for the suction side.

$$\min_{x \in \mathbb{R}^{23}} f(x) = \sum_{i=1}^{2*nb\_pnts} ||P_i(x) - T_i||^2 \tag{4.7}$$

where x are the design parameters of the optimisation problem.
This fitting algorithm solves a gradient-based optimisation problem. The derivatives of the airfoil profile w.r.t. the controlling parameters (Sec. 4.2) are computed with the reverse mode of AD. These derivatives are then given to the L-BFGS optimisation algorithm.

The Limited memory BFGS – L-BFGS

The optimiser used is a quasi-Newton method for solving unconstrained nonlinear minimization problems, the Limited Memory BFGS, or L-BFGS [115]. The L-BFGS approximates the inverse Hessian matrix of the objective function by a sequence of gradient vectors from previous iterations. Defined $f(x)$ as the cost function to be optimised and x the set of unknown variables (in this case the design parameters of the *proposed parametrisation*), at the k-th iteration of the L-BFGS the variables x_{k+1} are updated as shown in Eq. 4.8:

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k) \tag{4.8}$$

with H_k that is the approximation of the Hessian matrix of $f(x)$ at x_k. The *search direction* is represented by $H_k \nabla f(x_k)$ whereas α_k is the step-size whose value is set to 1 [115].

4.3.1 Optimisation results

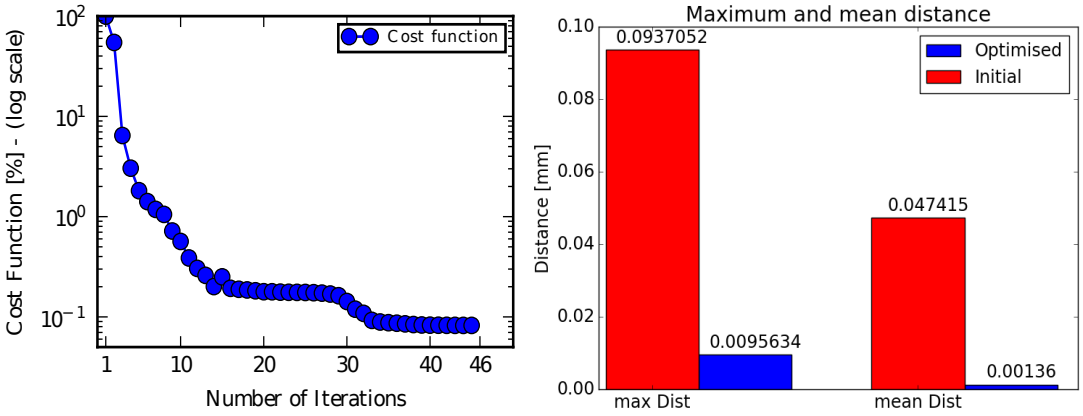


Figure 4.5: Left: optimisation convergence history in semi-logarithmic scale. Right: Mean and maximum distance between the two shape at the before and after the fitting problem.

Fig. 4.5 presents the optimisation history, which converges after 46 iterations and reduces the cost function by 99.9%. The maximum distance is reduced by factor 100 (from 0.09 mm to 0.0009 mm) whilst the mean distance is reduced from 0.047 mm to 0.00136 mm. Fig. 4.6 shows the initial and the optimised

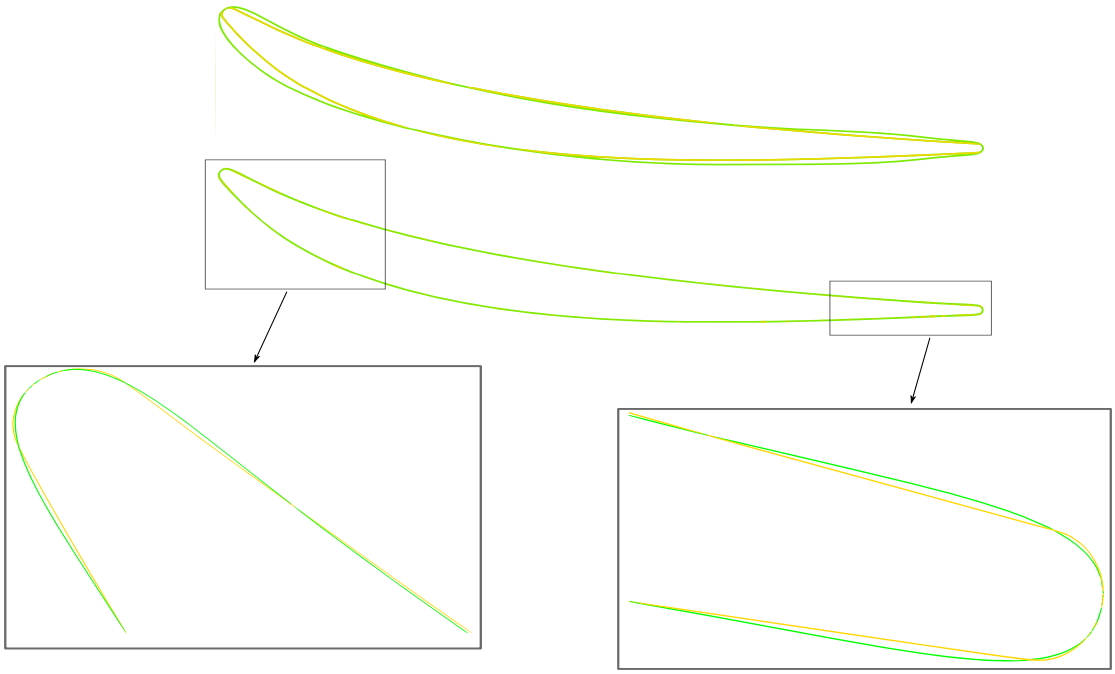


Figure 4.6: Top: initial section profile (green) and target profile (gold). Bottom: optimised 2D profile (green) and target profile (gold) with highlighted the differences between the two shapes at LE and TE.

configurations of the optimisation problem (the *proposed parametrisation* is the green one while the *target section* is in gold). The two profiles do not fit completely. The reason of this is that the *target section* is parametrised within a commercial CAD system by using circular arcs to control the leading edge and the trailing one (as explained in Sec. 4.1). The *proposed parametrisation* has been developed in order to avoid such construction (Sec. 4.2) and therefore can not reproduce exactly the baseline geometry. The minor differences are verified in the area of the LE and TE, as shown in Fig. 4.6.

4.4 Summary

This Chapter has investigated the possible approaches which can be used to parametrise an airfoil. The chosen approach (the "proposed parametrisation") considers the engineering parameters normally handled by the designer during the conception phase of the geometry. This suits the typical industrial workflow. The suction and pressure side of the airfoil are parametrised with two B-spline curves. This allows to guarantee the G2 continuity at the leading and trailing edge of the airfoil by imposing that, at these points, both sides (suction and pressure ones) have the same radius of curvature.
The aforementioned parametrisation has been automatically differentiated in reverse mode. This has allowed to define a reparametrisation tool which refits the existent baseline 2D section of the TUB compressor stator blade with the proposed parametrisation. The fitting algorithm solves a gradient-based optimisation where the derivatives of the cost function (total distance between the existent baseline geometry and the proposed parametrisation) w.r.t. the design parameters of the proposed parametrisation have been calculated with the reverse mode variant of the automatically differentiated OCCT. The cost function is almost nullified such that minor differences between the baseline and the optimised geometries are identified. These differences are due to the different parametrisation used to implement the two CAD models (the *proposed parametrisation* and the target one).

Chapter 5

Parametrisation of the test-cases

This chapter presents the feature tree of the automatically differentiated CAD models which are used to perform the aerodynamic shape optimisation of the test cases studied in this thesis: a cooling channel (optimisation results in Sec. 6.5) and a compressor stator blade (optimisation results in Sec. 7.6).
Sec. 5.1 explains how the test-cases have been identified. Sec. 5.2 introduces the first test case: the U-bend, a cooling channel for high-pressure turbine blades. Sec. 5.3 shows how a channel can be parametrised by using **OCCT**. Sec. 5.4 gives the algorithm used to parametrise the geometry of the test cases. Sec. 5.5 presents the parametrisation chosen for the U-bend. Sec. 5.6 provides the validation of the shape derivatives computed with the differentiated **OCCT** in both *traceless* forward mode and *trace* reverse mode. Sec. 5.7 introduces to the second test case studied: a coompressor stator blade. Sec. 5.8 explains how this test case has been parametrised and Sec. 5.9 shows the validation of the shape derivatives.

5.1 Why these test cases?

The goal of aerodynamic shape optimisation is to improve the performance of the product (e.g. engine aircraft, Fig. 5.1). The performance of the product is normally affected by a limited number of components. Therefore, designers are used to extract from the CAD model of the final product (e.g. an aircraft engine) the geometry of the components which are crucial to its performance (e.g. compressor stator blade), optimise the shape of this component and then reinsert the optimised shape in the overall CAD model (Fig. 5.1). For an aircraft engine, the crucial components are the blades of the stator/rotor of the turbomachines (i.e. compressors/turbines) and the pipes/channels which have the strongest impact on the thermodynamic cycle of the engine such as the cooling ones.

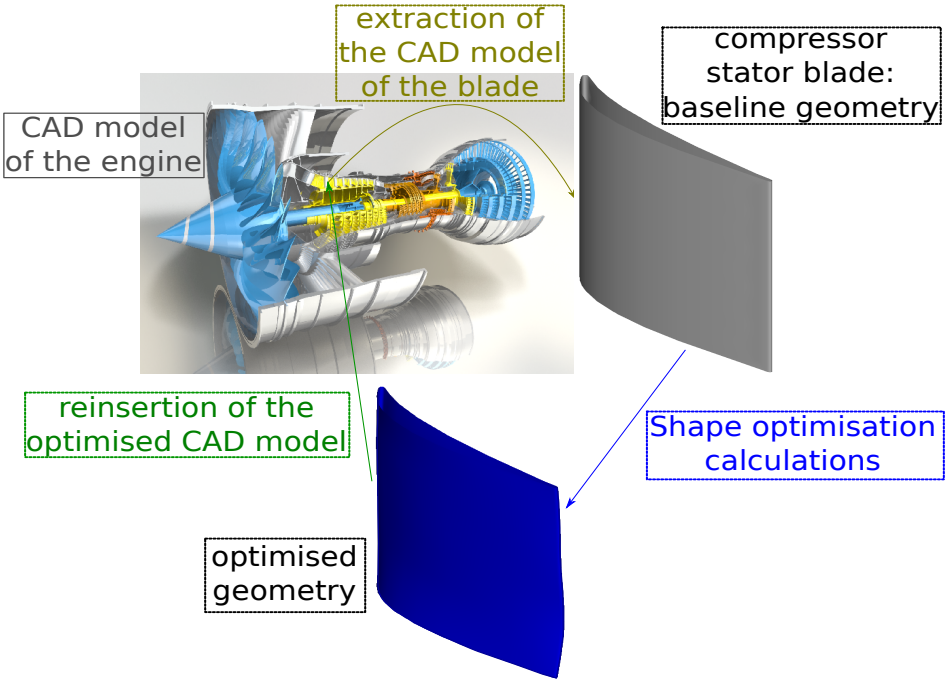


Figure 5.1: Shape optimsiation: design loop. Given the CAD model of the overall aircraft engine (Safran and OCC courtesy), the geometry of the blade of the compressor stator is extracted. This geometry is optimised and the geometry with improved performance is reinserted in the CAD model of the overall product.

The two test-cases investigated in this thesis are fixed by the EC project IODA[1]. The first test-case is the U-bend, which is a cooling channel for high pressure turbine blades. This channel has an important impact on the thermodynamic cycle of the engine, as explained in Sec. 5.2. The second test-case is the blade of the compressor stator installed at Technical University of Berlin (TUB) test-rig (Sec. 5.7). The settings used to perform the aerodynamic shape optimisation of these test-cases have been defined within the EC framework IODA and are used in this thesis to perform the shape optimisation of the automatically differentiated CAD models.

Another reason which justifies the choice of these test-cases is that they allow to study two engineering problems normally investigated in the R&D department of industries. The U-bend allows to investigate a flow problem which is typical of cooling serpentine. The compressor stator blade is an external fluid-dynamics problem which also prescribes the imposition of the assembly constraints. The respect of these constraints, which is important to straightforwardly reinsert the optimised geometry into the CAD model of the final product, has never been demonstrated before. Ch. 7 provides a detailed investigation about this topic.

[1] `https://ioda.sems.qmul.ac.uk/benchmarks/`

5.2 U-bend test case

5.2.1 Introduction

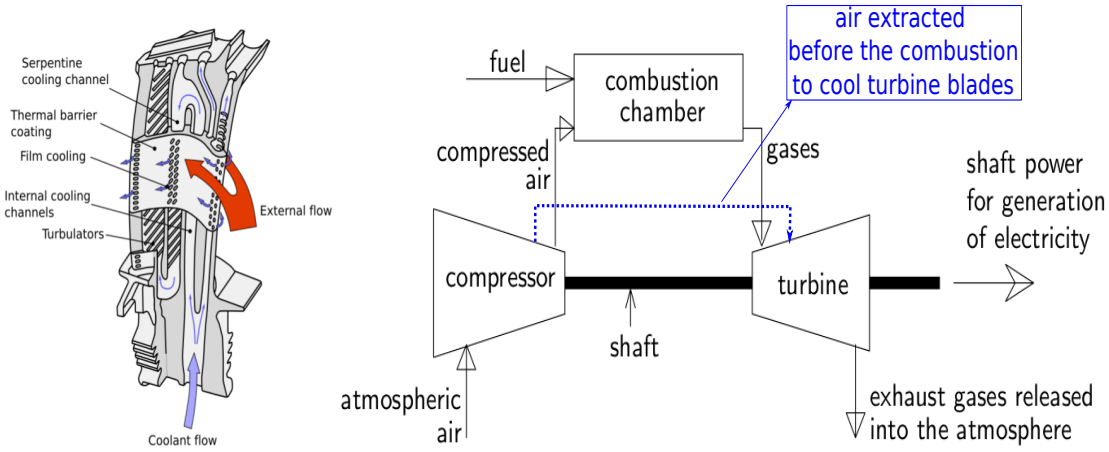


Figure 5.2: Cooling system of turbine blades [116], on the left. Flow diagram of a gas turbine (single shaft), on the right [117].

U-bend channels are normally used in industrial applications to turn the direction of the flow of 180 degrees. The U-bend[2] studied in this research is a cooling channel utilised in gas turbines to reduce the temperature of turbine blades overheated by the flow that is exiting by the combustion chamber (Fig. 5.2). The efficiency of the thermodynamic cycle of gas turbines increases with the maximum temperature reached in the chamber of combustion (i.e. the firing temperature). The cooling of turbine blades becomes crucial to enable a high firing temperature and therefore to let the gas turbine have high cyclic efficiency (i.e. high performance).

As shown in Fig. 5.2, cooling channels are usually defined by a single channel that traverses the span multiple times. The U-bends turn the flow by 180° near the tip and the bottom of the blade. The cooling fluid is pressurised air that is diverted from the compressor and conveyed to the turbine blade through its root such that the combustion chamber is bypassed (Fig. 5.2). The cooling fluid therefore is not used by the turbine to extract energy. This causes a reduction of the energy (i.e. the power) produced by the gas turbine. Such reduction can be limited by optimising (i.e. minimising) the total pressure losses between the inlet and the outlet of the cooling channel. Several experimental studies [118, 119] demonstrate that U-bends cause around the 25% of the pressure losses in such type of cooling

[2] `https://aboutflow.sems.qmul.ac.uk/content/events/munich2016/benchmark/testcase1/`

channels. Thus, the pressure losses between the inlet and the outlet of the U-bend will be defined as the cost function to be minimised. Further fluid-dynamic settings of the optimisation are given in Sec. 6.3. In this chapter, the author will discuss the geometry of the U-bend under investigation and the parametrisation chosen to perform the aerodynamic shape optimisation.

5.2.2 Geometry

The geometry of the test case under investigation is shown in Fig. 5.3. It consists of a semi-circular U-bend with a hydraulic diameter D = 0.075 mm. The hydraulic diameter is commonly used when handling flows in non-circular tubes and channels. Its formula[3] is (4 A / p), with A that is the area of the cross-section and p that is the "wet perimeter" of the cross-section. In this case, the cross-section is squared and, therefore, the hydraulic diameter coincides with the side of the square.

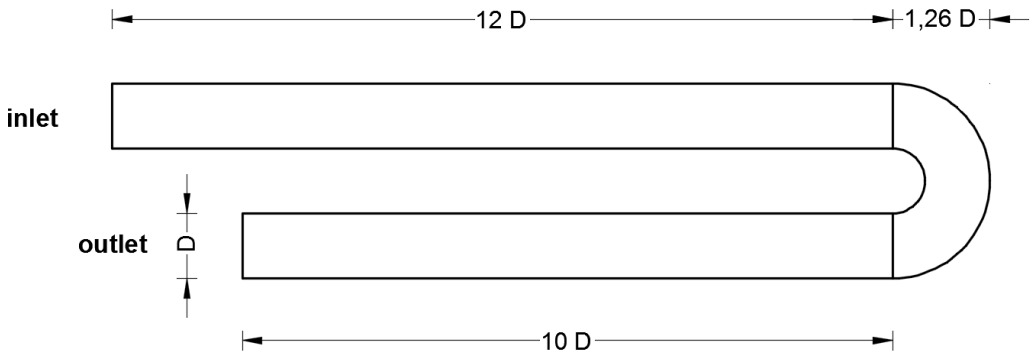


Figure 5.3: U-bend Dimensions.

5.3 Parametrisation of channels in OCCT

A first approach which can be used to parametrise a pipe in shape optimisation is by using NURBS surfaces, as proposed by other authors [5, 120]. This approach is not considered in this research work because, as explained in Ch. 2, this thesis aims to utilise parametric-based designs to aerodynamically optimise the shape. OCCT offers several approaches to design a channel with parametric CAD models. The most immediate approach is to define an initial 2D section (e.g. a circle,

[3] `https://neutrium.net/fluid_flow/hydraulic-diameter/`

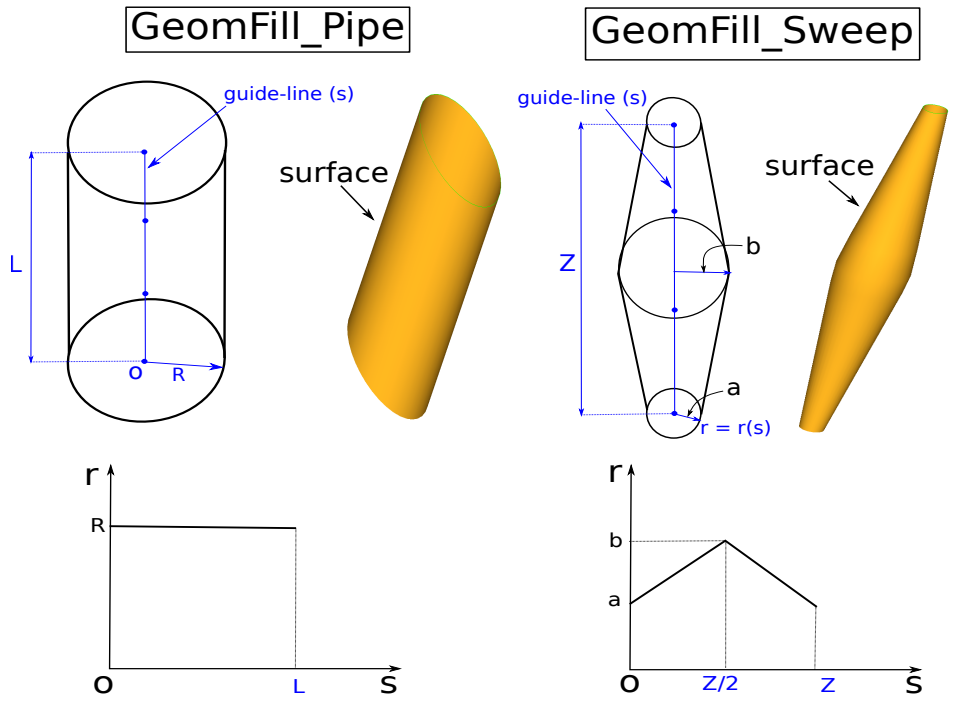


Figure 5.4: Possible ways of defining the parametric CAD model of a channel with **OCCT**. Left, (GeomFill_Pipe): the parameters of the channel are the radius of the circular section and parameters of the path-line. Right, (GeomFill_Sweep): the parameters of the channel is the function of radius r along the path-line. For both cases, the topology of the 2D section (circular) does not change along the path-line. On the bottom, the evolution of the radius along the path-line: r = r(s).

radius r) and a path-line along which the surface of the channel is calculated (Fig. 5.4). The design parameters of the optimisation would be the parameter of the 2D section (i.e. the radius of the circle) and the parameters controlling the path-line (i.e. the control points of the B-spline curve, Fig. 5.4). `BrepFill_Pipe` and `GeomFill_Pipe` are the reference classes provided by **OCCT** to perform such design. This design approach, that is very simple for an immediate implementation of the CAD model, is not suitable for shape optimisation because the values of the parameters that control the 2D section are kept constant along the path-line (Fig. 5.4). The implementation of the CAD model of a channel by using `GeomFill_Pipe` is shown in code listing A.1.

Another design approach (Fig. 5.4) is to specify a 2D section (ex. circle), a path-line (where s is the parameter of the parametric equation of this curve), and a law to drive the evolution of the parameters of the 2D section along the path-line (e.g. law of evolution of the radius of the circle along the path-line). The design parameters that control the CAD model of the channel are the parameters of the law of evo-

lution ($r = r(s)$) and the parameters of the path-line (the path-line is normally a B-spline curve). This design approach does not suit aerodynamic shape optimisation because, even if the 2D section can change along the path-line, the topology of the 2D section remains the same (i.e. all the 2D sections of the channel in Fig. 5.4 are circles). This approach can be implemented by using the **OCCT** class `GeomFill_Sweep`. The implementation of this parametrisation approach is shown in the code listing A.2. The last design approach provided by **OCCT** is based

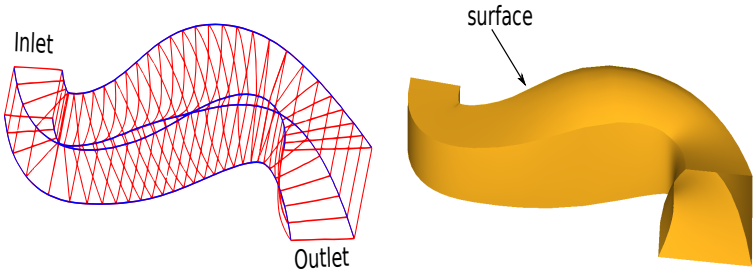


Figure 5.5: Example of surface constructed by using `BRepOffsetAPI_ThruSections` for a given set of 2D sections (shown in red). On the left: variation of the section from the inlet to the outlet. On the right, the final surface.

on the reference class `BRepOffsetAPI_ThruSections`. This class implements an algorithm that calculates a B-spline surface based on constraints. The constraints are defined as a set of 2D sections. Fig. 5.5 shows a channel retrieved from the approximation of a set of sections by utilising `BRepOffsetAPI_ThruSections`. The main advantage of this design approach w.r.t. the previous ones is that the 2D sections can be defined independently one from each other. On the contrary of the previous approaches, the surface of the channel can therefore adapt its shape based on the particular flow conditions. This is important to increase the possibilities to identify the shape with best performance.
The Code listing A.3 presents the implementation of the CAD model of a channel by utilising `BRepOffsetAPI_ThruSections`. The design approach used to implement the parametrisation of the U-bend is based on `BRepOffsetAPI_ThruSections`, as explained in Sec. 5.5.

5.4 BRepOffsetAPI_ThruSections

`BRepOffsetAPI_ThruSections` is the algorithm implemented in **OCCT** to compute the B-spline surface approximating a set of sections previously defined. This algorithm solves an optimisation problem with constraints where the constraints

are represented by the set of sections. To simplify the explication, the author will refer to a problem where a B-spline curve $C(t)$ is constrained to pass through a set of points $(P_{c,i})$ with $i = 0 \dots N_c$, where N_c represents the number of points to be approximated.
The optimisation problem [121] solved by `BRepOffsetAPI_ThruSections` minimises the cost function L defined as in Eq. 5.1:

$$L = L_0 + f(f_1 L_1 + f_2 L_2 + f_3 L_3) \tag{5.1}$$

where L_0 represents the total distance between the set of points $P_{c,i}$ and the B-spline curve $C(t_i)$, Eq. 5.2:

$$L_0 = \sum_{i=0}^{N_c} ||C(t_i) - P_{c,i}||^2 \tag{5.2}$$

and the functions (L_1, L_2, L_3) defined as the *quality criteria* of the optimisation problem are given by Eq. 5.3:

$$\begin{cases} L_1 = \int_a^b C_1^2(t)dt \\ L_2 = \int_a^b C_2^2(t)dt \\ L_3 = \int_a^b C_3^2(t)dt \end{cases} \quad \text{with} \quad \begin{cases} ||C_1||^2 = \left(\frac{S_C}{b-a}\right)^2 \\ ||C_2||^2 = \rho^2(t) = ||\ddot{C}(t)||^2 \\ ||C_3(t)||^2 = \rho^4(t) + \dot{\rho}^2(t) + \rho^2(t)\Omega^2(t) \end{cases} \tag{5.3}$$

where $[a, b]$ is the interval where $C(t)$ is defined, S_c is the length of the curve $C(t)$, ρ represents the curvature of the curve $C(t)$ and $\Omega(t)$ is the torsion of the curve $C(t)$.
Each of the functions of the *quality criteria* is weighted with a coefficient f_j $(j = 1 \dots 3)$ and the overall *quality criteria* expression $(f_1 L_1 + f_2 L_2 + f_3 L_3)$ is weighted with the coefficient f. Moreover, the optimisation problem expects that the curve $C(t)$ is *C2 continuous* in all the points $P_{c,i}$ $(i = 1 \dots N_c)$.
The curve generated by this algorithm is function of the number of points/constraints considered, i.e. C = f(N_c). It is possible to evaluate the minimum number of points $N_{c,min}$ necessary to "analytically" determine the final B-spline curve, (i.e. to determine a curve which is independent of the number of constraints/-points). $N_{c,min}$ is identified as follows:

- an initial number of points $N_{c,1}$ is set and the relative B-spline curve $C_1(N_{c,1})$ is generated.
- A new number of points is defined, $N_{c,2} = N_{c,1} + 1$ and the relative curve

$C_2(N_{c,2})$ is generated,

- if C_1 and C_2 are coincident, $N_{c,min} = N_1$.
- if the algorithm has not generated two coincident curves, the loop proceeds by adding another constraint/point.

5.4.1 Design approach based on `BRepOffsetAPI_ThruSections`

For both test cases studied in this research work, the author has developed a feature tree which takes as input the design parameters of the CAD model and provides as output the final CAD model. The CAD model (Ubend: Fig. 5.7, compressor stator blade: Fig. 5.15) is implemented based on the following steps:

1. a 2D section governed by a set of parameters is defined.
2. For every parameter a law of evolution that drives the parameter along a path-line (the *pathline*) is set.
3. Per every plane orthogonal to the *pathline*, the 2D section is constructed by retrieving from the laws of evolution the values of the parameters.
4. All the sections are used as constraints by `BRepOffsetAPI_ThruSections` to compute the B-spline surface.

5.5 U-bend parametrisation

The U-bend geometry consists of three main parts (Fig. 5.7): (i) the inlet pipe, (ii) the outlet pipe and (iii) the U-turn. In this thesis, the optPart is the part of the geometry to be optimised. The optPart of the U-bend is the U-turn. The test case requires to respect the G1 continuity between the inlet/outlet leg and the U-turn.
The U-turn parametrisation is based on the design approach explained in Sec. 5.4, which starts with the definition of the parameters controlling the 2D section.

5.5.1 2D section

Every side of the 2D section is a cubic Bezier curve, which are suitable to shape optimisation when considering a reduced number of control points [57, 58]. Each Bezier curve consists of 4 control points and shares the first and the last control point with the previous and the following Bezier curve. Therefore, the 2D section

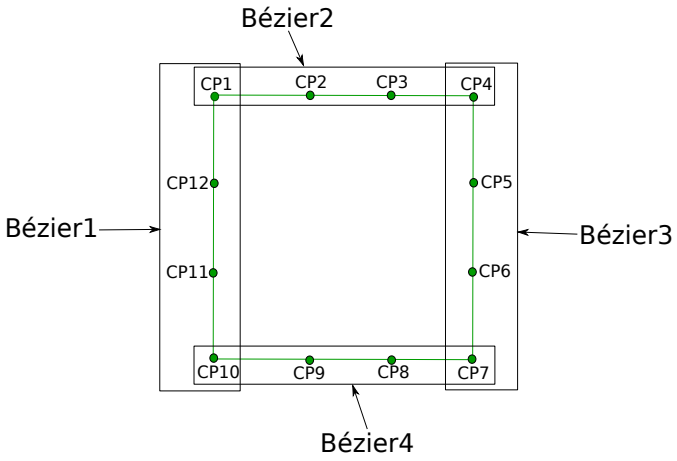


Figure 5.6: 2D section of the U-bend.

has in total 12 control points (i.e. 24 design parameters), as shown in Fig. 5.6. This choice allows the section to assume a wide variety of shapes that the optimiser could impose during the flow optimisation.

5.5.2 3D Building and imposed constraints

The main elements of the CAD model of the U-bend are shown in Fig. 5.7. As explained previously in this section, each parameter of the 2D section is controlled by a law of evolution along a 3D curve, the *pathline*. Every law is a B-Spline curve. The number of control points per every law of evolution determines the refinement of the design space. It has been demonstrated that gradient-based methods are much more convenient w.r.t. gradient-free ones when the parametrisation consists of 50-100 design parameters [15]. Moreover, for the parametrising bends, 50-100 design parameters have been demonstrated to effectively control the shape [122, 123].

In this research work, every law of evolution is defined as a B-Spline curve that consists of 8 control points. The first two and the last two control points of every law are fixed in order to assure the tangency constraint with the inlet and outlet legs of the U-bend (Sec. 5.2). The total number of design parameters is therefore 96 (24 (number of parameters of the 2D section) $\cdot$ 4). The optimisation results provided by this design space are compared to the ones given by a NURBS-based parametrisation (the NSPCC approach), which parametrises the shape with a refined design space (hundreds of control points).

The steps performed to construct the optPart (Sec. 5.2) of the U-bend are the followings:

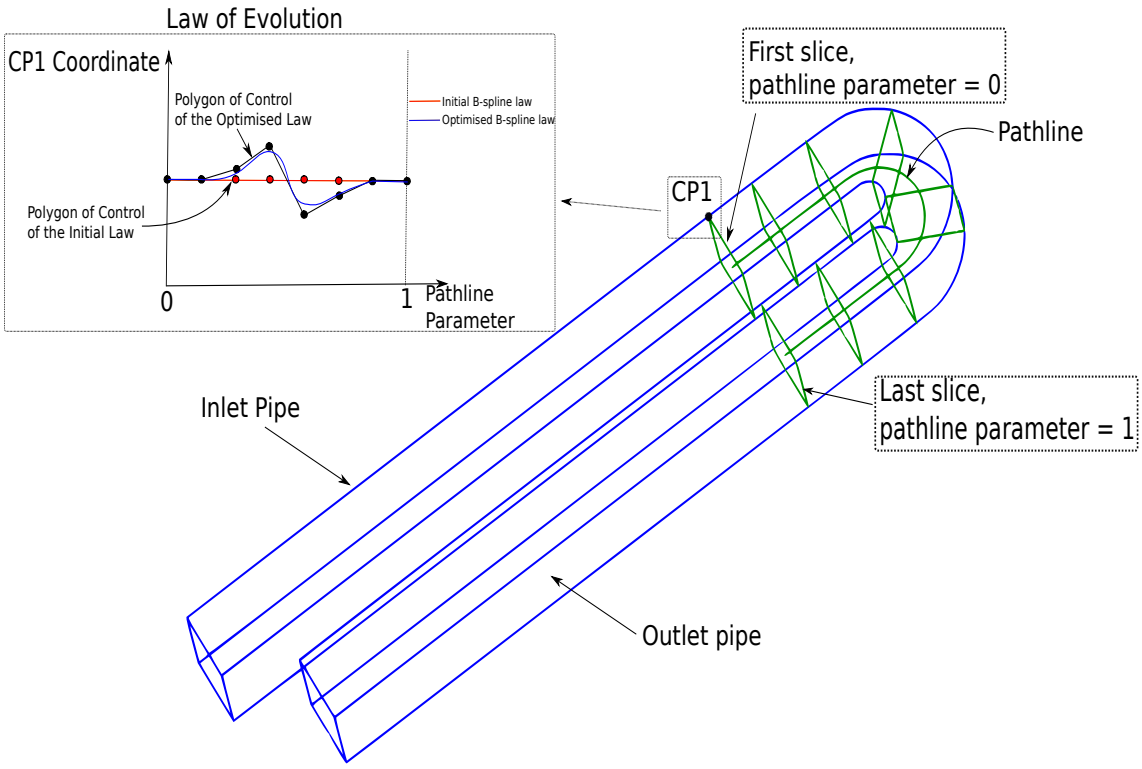


Figure 5.7: Parametrisation of the U-bend.

1. the 2D section (i.e. the slice) is defined.
2. The *pathline* is determined.
3. The laws of evolution are built.
4. Given a plane orthogonal to the *pathline* in a point, the slice relative to this plane is constructed based on the laws of evolution.
5. `BRepOffsetAPI_ThruSections` allows to compute the B-spline surface of the optPart (Fig. 5.7).

The number of slices to be approximated is identified by the procedure explained in Sec. 5.4. For the U-bend, $N_{c,min}$ is 68. This value is rounded to 70 to easily remember it.
The design explained above is applied to the optPart of the U-bend, which is the U-turn (Sec. 5.2). The inlet and the outlet legs of the channel are constructed by extruding the first and the last sections of the U-turn with the reference class `BRepPrimAPI_MakePrism`. The final shape of the U-bend is shown in Fig. 5.7.

5.5.3 U-bend: implementation of the CAD model

In this subsection, the author introduces the feature tree of the CAD model of the U-bend, which is implemented by the C++ function `constructUbend`. This feature takes as input a set of `Standard_Real` (i.e. the design parameters of the optimisation) and provides as output the geometry of the U-bend.
The author defines the *geometric space* as the space where the CAD model is constructed and the *parametric space* as the space where the laws of evolution are implemented. After the definition of the laws of evolution (*parametric space*), the first step is the construction of the *pathline* by using the `Geom_BSplineCurve` reference class. This curve identifies the optPart of the CAD model. A set of points is uniformly distributed on the *pathline* by using the `GCPnts_UniformAbscissa` reference class. The number of points is determined by the number of 2D sections to be approximated ("nbSlices"). Then, the planes orthogonal to the *pathline* passing through the distributed points are created (`Geom_Plane` reference class). Per every plane, the coordinates of the 12 points that will control the 2D section are computed in the *parametric space* (reference class `GeomAPI_IntCS`) and created on the planes. These 12 points are used to construct the 4 edges (i.e. sides) of the 2D section (`BRepBuilderAPI_MakeEdge`). These 4 edges are then processed by `BRepBuilderAPI_MakeWire` to construct a closed wire (i.e. the 2D section, reference class `TopoDS_Wire`). All the wires are added to the approximation algorithm `BRepOffsetAPI_ThruSections` that computes the final surface.
A simplified implementation of the feature tree is reported in the flow chart shown in Fig. 5.8. Code listing B.1 presents the C++ function `constructUbend`.

5.6 Validation of the shape derivatives of the U-bend

The validation of the derivatives and the analysis of the computational performance (run-time ratios and the memory requirements, Subsec. 5.6.1) have been conducted by the collaborator who was responsible for the differentiation of the CAD kernel in the IODA project, Mladen Banovic
The shape derivatives computed with the differentiated **OCCT** in *traceless* forward mode are compared to finite differences. Fig. 5.9 shows, as a representative example, the shape derivatives w.r.t. one design parameter. The overall magnitude plots illustrate that these results coincide to a very high extent. Also the quantitative comparison between AD and FD for the same design parameter shows

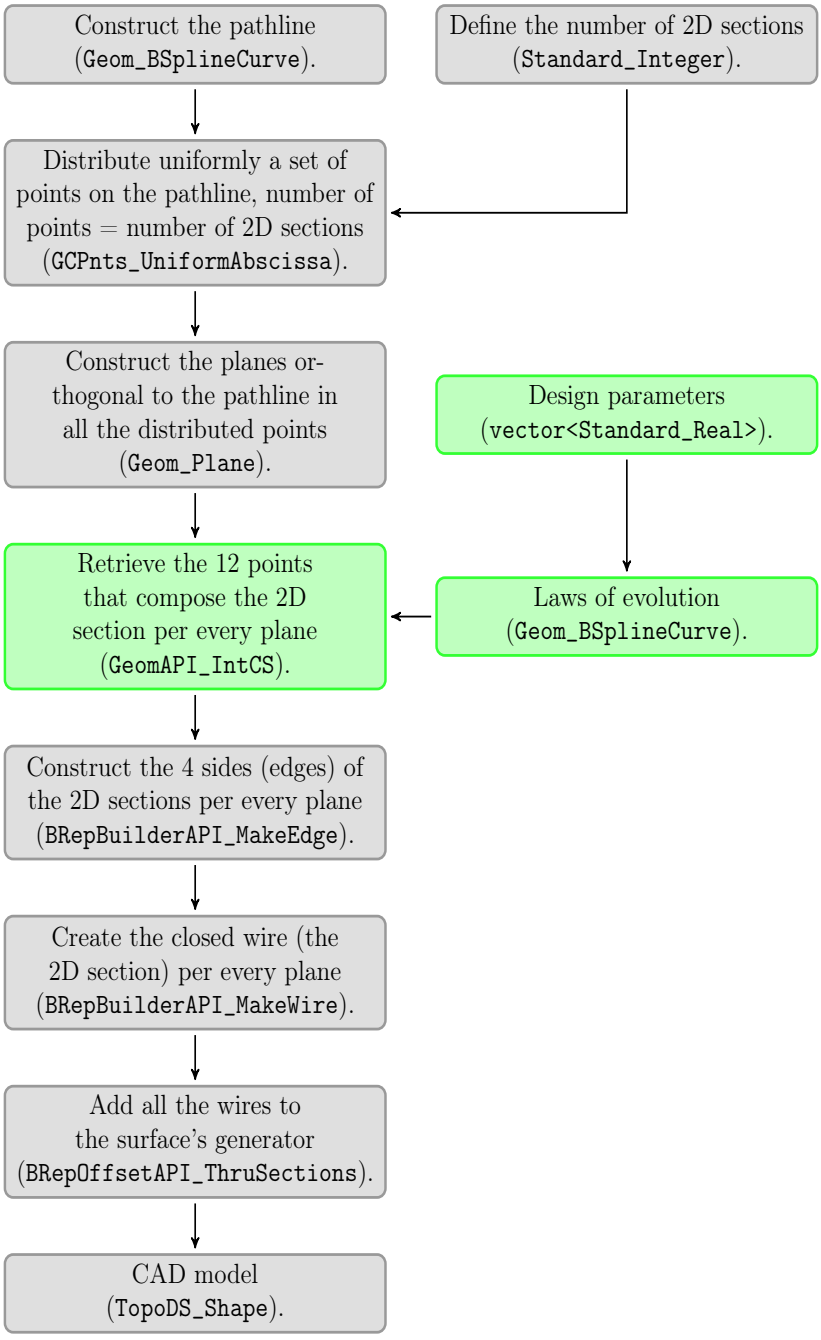


Figure 5.8: Flow chart of `constructUbend`. The steps in gray are performed in the *geometric space*. The steps in green are executed in the *parametric space*. Per every step of the flow chart, the **OCCT** reference class is mentioned.

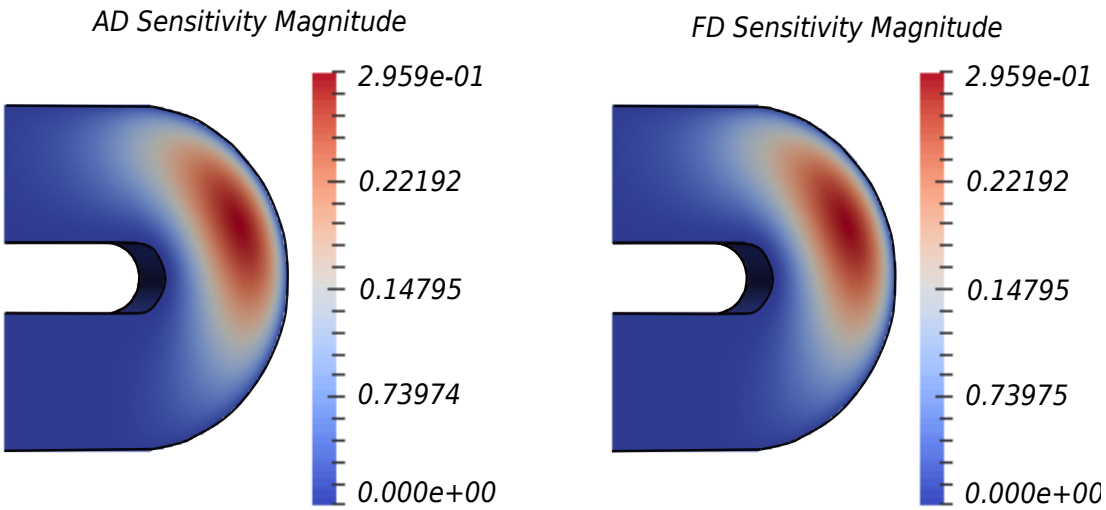


Figure 5.9: U-turn derivatives evaluated by AD (left) and finite differences (right), as reported by Banovic et al. [105].

AD value	FD value	Abs. difference
0.00038436	0.00038426	1.02e-07
0.06339189	0.06339125	6.41e-07
0.15615249	0.15614906	3.43e-06
0.12874039	0.12873815	2.24e-06
⋮	⋮	⋮
0.27459387	0.27458089	1.30e-05*

*Maximal difference.

Table 5.1: AD and FD values comparison for several U-bend point coordinates.

mutual agreement (Table 5.1). The Taylor test (Eq. 5.4) is used to verify the shape derivatives computed with finite differences:

$$f(x+h) - f(x) - h\frac{\partial f}{\partial x}(x) = \mathcal{O}(h^2) \tag{5.4}$$

The Taylor test has been performed on an arbitrary number of surface mesh point coordinates (eight) with a step size $h[10^{-1}, 10^{-10}]$. This test allows to compare the error (left side of Eq. 5.4) to the theoretical convergence rate of h^2. Fig. 5.18 shows that for these eight points the convergence rate is faster then the one of h^2. This allow to verify the correctness of the computed derivatives.

The derivatives computed with the *trace-based* variants are validated against the *traceless* forward mode. The results presented in Table 5.2 show some small disparity (close to machine precision) between the gradients. This is due to the differences between some overloaded operators of ADOL-C in the *trace-based* and

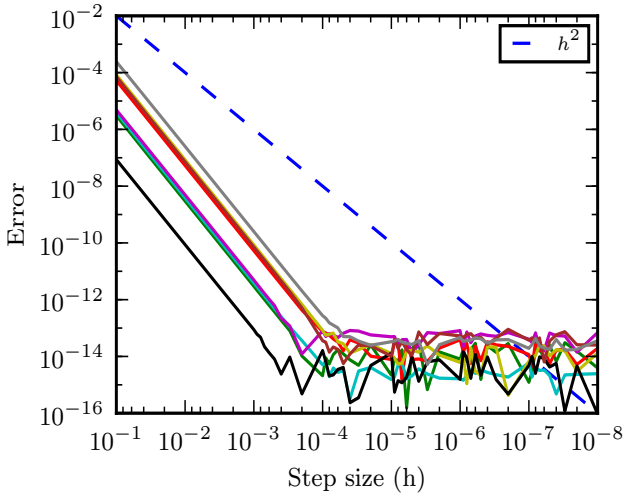


Figure 5.10: Taylor test overview for eight surface mesh points coordiantes [105].

AD Traceless-forward gradient	AD Trace-based forward gradient	Abs. difference
1.03293863915149e-01	1.03293863915283e-01	1.34e-13
2.36707086987917e-01	2.36707086987913e-01	4.00e-15
1.28682761975210e-01	1.28682761975225e-01	1.50e-14
1.25652442670046e-02	1.25652442669980e-02	6.60e-15
⋮	⋮	⋮
2.24699593280905e-01	2.24699593281250e-01	3.45e-13*

*Maximal difference.

Table 5.2: AD Traceless-forward and AD Trace-based forward gradient comparison for several U-bend point coordinates.

traceless options. Not only the derivative calculation but also the primal evaluation is affected in the same order of magnitude. The differences between the *trace-based* and the *traceless* variants are therefore small enough not to yield radically different CAD models, and hence both can be used equally.

5.6.1 Run-time ratio and memory requirements

The computational performance (run-time ratio and memory requirements) of the automatically differentiated parametric CAD model (both *traceless* forward mode and *trace-based* forward/reverse mode) is measured by computing $df(x_j)/dx_j$ for all the design parameters of the U-bend x_j ($j = 1 \dots 96$), where $f(x)$ is expressed in Eq. 5.5. The run-time ratio of the finite differences is also reported together with the AD run-time ratios. $f(x)$ computes the "total distance" between the original U-bend geometry P (generated with the initial set of design parameters)

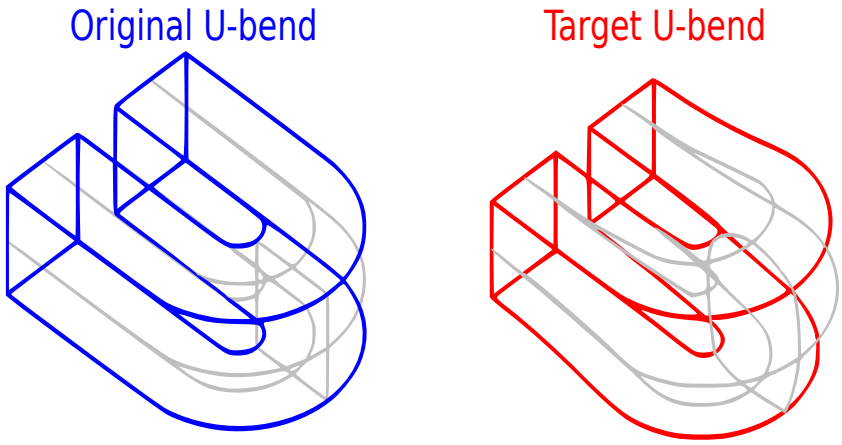


Figure 5.11: Test case to evaluate the computational performance of the differentiated parametrisation of the U-bend: original U-bend (left) and target U-bend (right) (reported by Banovic et al. [105]).

and a target geometry T:

$$f(x) = \sum_{i=1}^{12000} ||P_i(x) - T_i||^2 \quad x \in \mathbb{R}^{96} \tag{5.5}$$

where i is the index of one of 12000 sampling points distributed uniformly over the surface, $P_i(x)$ and T_i are the points on the original and target (perturbed) surfaces respectively. The verification proceeds as follows:

1. construct two U-bends by providing to the feature tree explained in Subsec. 5.5.3 two different set of design parameters: $x_{j,or}$ and $x_{j,tar}$. $x_{j,or}$ allows to generate the U-bend geometry with the original dimensions and $x_{j,tar}$ builds the target U-bend geometry (Fig. 5.11).

2. Sample both final B-spline surfaces (original and target) with 12K pairs of (u_i, v_i) parametric coordinates. These parametric coordinates are later used in B-spline algorithms to evaluate the corresponding three-dimensional points $P_i(x)$ and T_i.

3. Define the objective function as in Eq. 5.5.

4. Declare the original set of design parameters ($x_{j,or}$) as independent variables of the system.

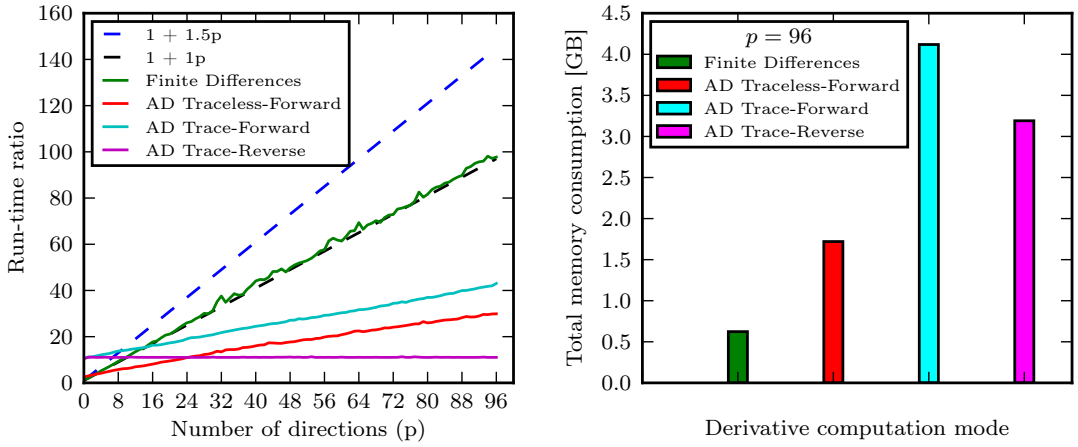


Figure 5.12: Overview of run-time ratios (left) and total memory requirements (right) for U-bend example, as reported by Banovic et al. [105].

5. Compute the derivatives of the cost function in Eq. 5.5 w.r.t. the design parameters $x_{j,or}$ (i.e. w.r.t. the design parameters of the original U-bend geometry P).

The tests are performed with all three differentiation modes of ADOL-C (the *traceless* forward mode and the *trace-based* forward and reverse modes). The measurements related to the *trace-based* modes include the time for generating the trace.

The run-time ratio is the ratio between the time taken by the computation of the derivatives by using the *FD* / AD *traceless-forward* / AD *trace-forward* / AD *trace-reverse* and the time necessary for the function ("primal") evaluation. Fig. 5.12 shows both the run-time ratios for all AD modes and for FD approach (on the left) and the memory requirements on the right. As expected, the FD computational cost scales linearly with the number of design directions/parameters p. AD trace-forward mode reproduces the same behaviour of the AD traceless-forward mode but the starting point of the first is shifted w.r.t. the latter. This shift is due to the time required by the trace-forward mode to perform the first sweep of the function (i.e. to tape the function `constructUbend`, Sec. 3.2). For the AD trace-reverse mode, the run-time ratio is independent of the number of design parameters. As explained in Sec. 3.2, the run-time ratio of the AD trace-reverse mode scales with the number of cost functions (single cost function in this case) and is independent of the number of design parameters.

The FD memory consumption represents also the space occupied by the primal (i.e. OCCT original sources). As expected, the AD *trace-reverse* mode requires

much more memory w.r.t. the original sources. In this case, the ratio between the AD *trace-reverse* and the primal is six.

5.7 TUB TurboLab Stator Blade

5.7.1 Introduction

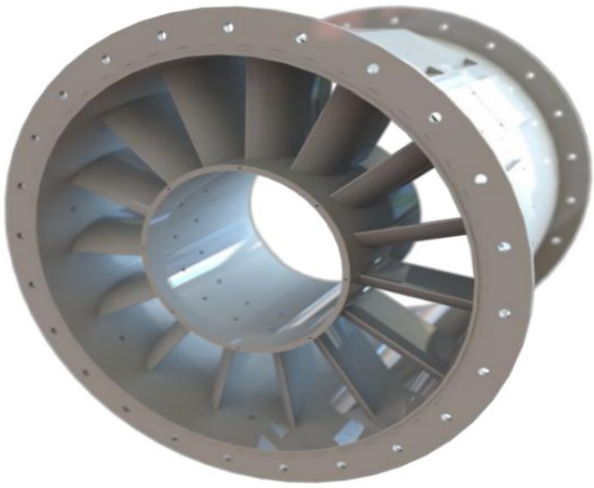


Figure 5.13: The "TurboLab Stator" in the measurement rig of the Technical University of Berlin, on the left.

The TUB TurboLab Stator (Fig. 5.13) is a compressor stator which has been investigated in a measurement rig at the Technical University of Berlin (TUB). It is a typical turbomachinery optimisation test case that is used to turn the incoming flow by 42°. It is one of the benchmark test cases of the shape optimisation workshop organised within the About Flow project[4]. The geometry of the stator blade has to be optimised in order to minimise the pressure losses between the inlet and the outlet of the stator. Further fluid-dynamics settings are provided in Sec. 6.8. In this chapter, we will analyse the geometry of the test case and explain the parametrisation chosen to perform the shape optimisation.

5.7.2 Geometry

The About flow website provides the baseline CAD geometry stored in three CAD exchange formats: IGES, STEP and parasolid. The geometric constraints of the

[4] `http://aboutflow.sems.qmul.ac.uk/events/munich2016/benchmark/testcase3/`

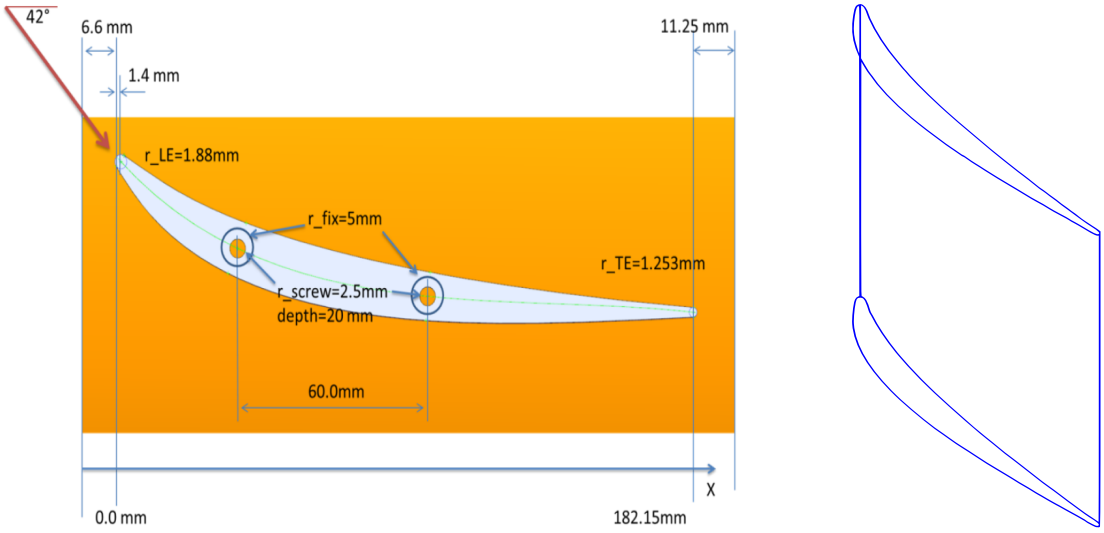


Figure 5.14: The geometric constraints of this test case.

TUB test case (Fig. 5.14) strongly influence the final optimised shape. The test case prescribes the following geometric constraints on the compressor blade:

1. fixed number of 15 blades such that the flow optimisation of a single blade with periodic boundary conditions is considered;
2. minimum radius of 1 mm at the leading and the trailing edge;
3. thickness distribution which reaches at least 8 mm value for any 2D profile of the blade;
4. minimum thickness near the hub and the shroud to accommodate the four mounting bolts (10 mm);
5. constant axial chord length (187.5 mm).

The blade is parametrised in **OCCT** such that all constraints except the assembly ones are explicitly embedded in the parametrisation.

5.8 Parametric CAD model of the TU Berlin Stator

Compressor blades are designed starting from the parametrisation of the 2D profile. Sec. 4.2 provides the parametrisation chosen for this 2D profile. This

parametrisation (Fig. 5.15) consists of the camber-line to control the inlet/outlet angle of the blade, a symmetric thickness distribution normal to camber-line and the LE/TE radius of curvature. In total, the number of design parameters controlling the 2D profile are 23.

5.8.1 3D blade building

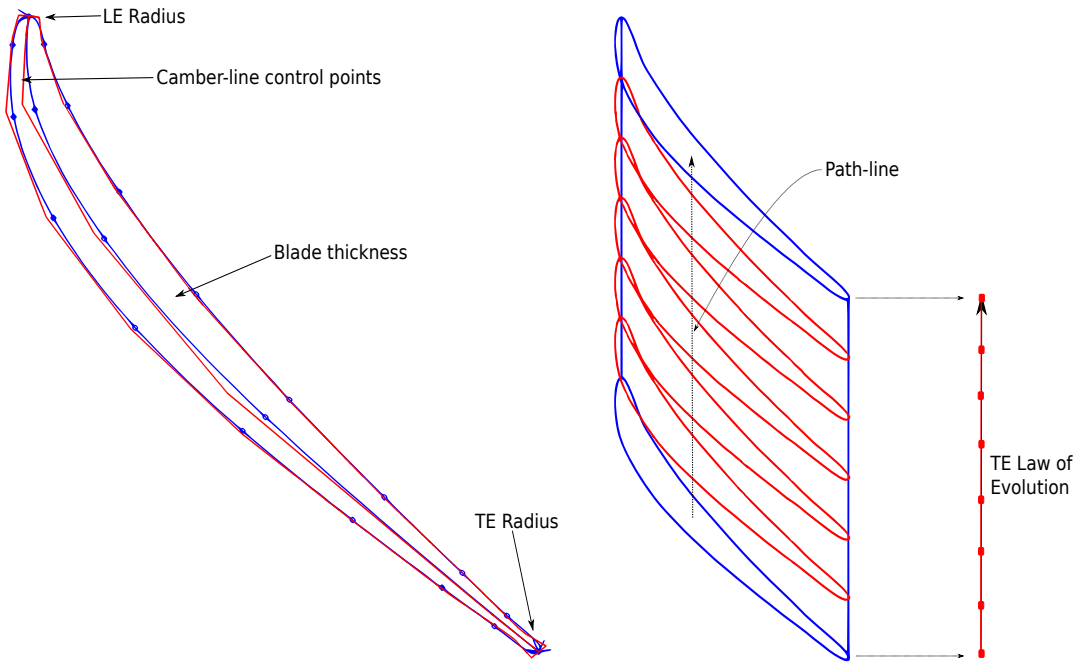


Figure 5.15: 2D Section Parameters of the Blade (left). Blade skeleton and TE law of evolution in the 3D domain (right).

The same fundamental feature tree as for the U-Bend (Subsec. 5.5.2) is adopted. Every parameter of the 2D section of the TUB is driven along the *pathline* by a law of evolution (Fig. 5.15). Every law is a B-Spline curve. The number of control points per law of evolution is fixed to 8. This is a number normally chosen by designers to perform the shape optimisation of 3D blades [75]. Based on the procedure explained in Subsec. 5.4, the total number of slices to be considered is 50.
The total number of design parameters is therefore 184: 23 (parameters of the 2D section) $\times$ 8. The results provided by this design space will be compared in Subsec. 7.7.2 to the ones given by an enlarged parametrisation (Subsec. 7.7.1), which controls the 2D profile with an almost doubled set of parameters (41 design parameters) and considers 16 control points per law of evolution.

AD value	FD value	Abs. difference
-3.9664798813e-07	-3.9664715956e-07	8.29e-13
6.4515277484e-07	6.4515282006e-07	4.52e-14
-8.2421051717e-06	-8.2420958947e-06	9.28e-12
-3.0079939641e-05	-3.0079916336e-05	2.33e-11
⋮	⋮	⋮
2.5489612529e-03	2.5489583777e-03	2.88e-09*

*Maximal difference.

Table 5.3: AD and FD values comparison for several blade point coordinates.

5.8.2 TUB: implementation of the CAD model

In this subsection, the author introduces the feature tree of the CAD model of the compressor stator blade, which is implemented by the C++ function `constructTUB`. This feature tree is developed such that it takes as input the design parameters of the CAD model and provides as output the geometry of the blade.
The flow chart of the TUB feature tree function is shown in Fig. 5.16. The function `constructTUB`, which consists of 900 lines of code, is presented in Appendix B, code listing B.2. The main difference with the implementation of the `constructUbend` (Fig. 5.8) relies on the definition of the 2D section. For every plane orthogonal to the *pathline*, `constructTUB` retrieves from the laws of evolution the coordinates of the control points of the camberline, the suction and the pressure sides. After having constructed the 2D sections, `BRepOffsetAPI_ThruSections` computes the final surface of the blade.

5.9 Validation of the shape derivatives of the blade

The validation of the derivatives and the analysis of the computational performance (run-time ratios and the memory requirements, Subsec. 5.9.1) have been conducted by the collaborator who was responsible for the differentiation of the CAD kernel in the IODA project, Mladen Banovic.
The shape derivatives computed with the *forward mode* (Sec. 3.2) are compared to finite differences. As a representative example, Fig. 5.9 shows the shape derivatives w.r.t. one design parameter. The overall magnitude plots illustrate that these results coincide to a very high extent. The quantitative comparison between AD and FD for the same design parameter shows mutual agreement (Table 5.3). The Taylor test (Eq. 5.4) is used to verify the shape derivatives computed with finite differences. It has been performed on an arbitrary number of surface mesh point

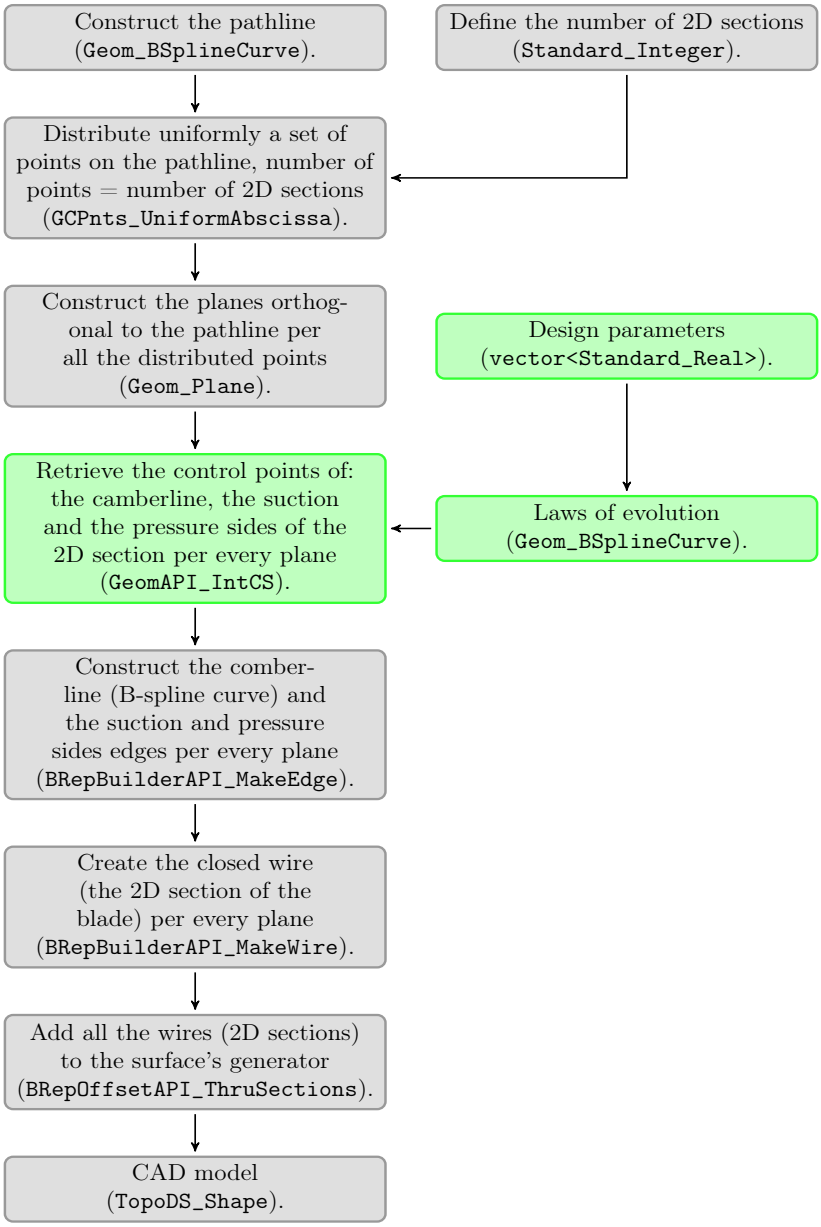


Figure 5.16: Flow chart of `ConstructTUB`. The steps in green are performed in the *geometric space*. The steps in green are executed in the *parametric space*. Per every step of the flow chart, the **OCCT** reference class is mentioned.

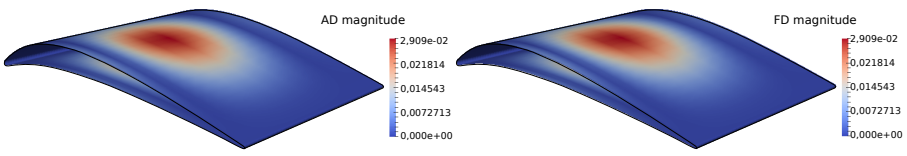


Figure 5.17: TUB stator blade derivatives evaluated by AD (left) and finite differences (right), as reported by Mykhaskiv et al. [85].

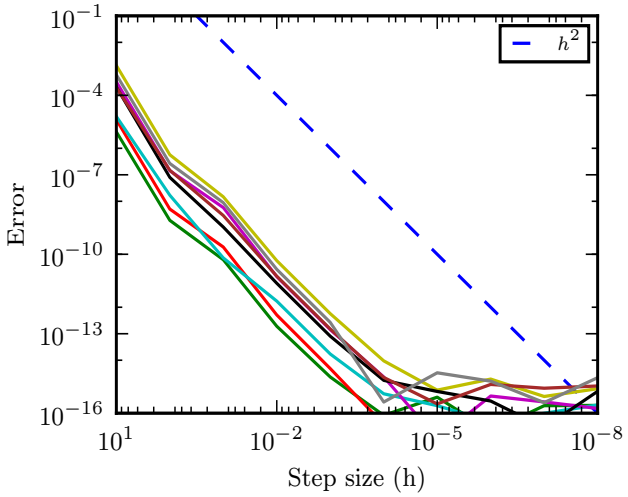


Figure 5.18: Taylor test overview for eight surface mesh points coordiantes [124].

AD Traceless-forward gradient	AD Trace-based forward gradient	Abs. difference
1.222314947336750e-05	1.222314947336682e-05	6.81e-19
-2.675251117738395e-06	-2.675251117738477e-06	8.17e-20
-6.058242522306152e-04	-6.058242522306268e-04	1.16e-17
-4.452581416338372e-07	-4.452581416338406e-07	3.44e-21
⋮	⋮	⋮
1.762384886025022e-03	1.762384886024947e-03	7.50e-17*

*Maximal difference.

Table 5.4: AD Traceless-forward and AD Trace-based forward gradient comparison for several TUB point coordinates.

coordinates with a step size h$[10^1, 10^{10}]$. This test allows to compare the error (left side of Eq. 5.4) to the theoretical convergence rate of h^2. Fig. 5.18 demonstrates that the convergence rate is faster then h^2. This verifies the correctness of the computed derivatives.

The derivatives computed with the *trace-based* variants are validated against the *traceless* forward mode. The results presented in Table 5.4 show some small disparity between the gradients. This disparity is due to the differences between some overloaded operators of ADOL-C in the *trace-based* and *traceless* options. Not only the derivative calculation but also the primal evaluation is affected in the same order of magnitude. The differences between the *trace-based* and the *traceless* variants are therefore small enough (i.e. below machine precision) to yield the to the same CAD model, and hence both can be used equally.

5.9.1 Run-time ratio and memory requirements

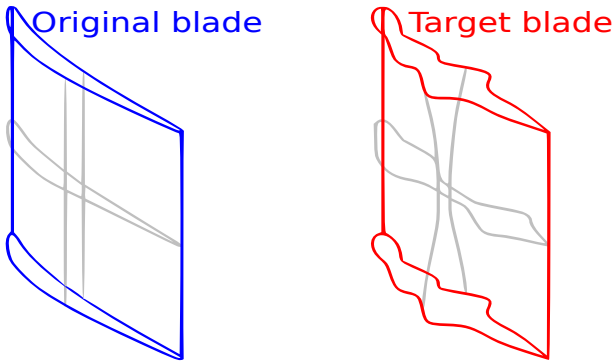


Figure 5.19: Test case to evaluate the performance of the differentiated parametrisation of the compressor stator blade: original blade (left) and target blade (right), as reported by Mykhaskiv et al. [85].

Similarly to the U-bend (Subsec. 5.6.1), it is necessary to define a function $f(x)$ w.r.t. the derivatives can be computed. In this thesis, given the target geometry shown in Fig. 5.19, the function $f(x)$ is computed as in Eq. 5.6:

$$f(x) = \sum_{i=1}^{1800} ||P_i(x) - T_i||^2 \quad x \in \mathbb{R}^{184} \tag{5.6}$$

where $P_i(x)$ and T_i are the points distributed uniformly on the original and target surfaces, respectively. $f(x)$ is utilised to analyse the run-time ratio and memory requirements of the automatically differentiated parametric CAD model (both *traceless* forward mode and *trace-based* forward/reverse mode) by calculating the derivatives of $f(x)$ w.r.t. the design parameters of the original blade.
The performance of the differentiated OCCT sources is analysed and compared to the original sources. The time required for the evaluation of the objective function and the corresponding derivatives' calculation is measured. The tests are performed with all three differentiation modes of ADOL-C (the *traceless* forward mode and the *trace-based* forward and reverse modes). The measurements related to the *trace-based* modes include the time for generating the trace. The run-time ratios for all the p-directions (i.e. number of design parameters, 184) and the total memory requirements w.r.t. the maximal number of directions are shown in Fig. 5.20. The run-time ratio of the FD approach is also reported together with the AD run-time ratios.

As shown in Fig. 5.20, the AD *trace-reverse* mode is more expensive in terms of memory consumption then the FD approach, but it is consistently faster particu-

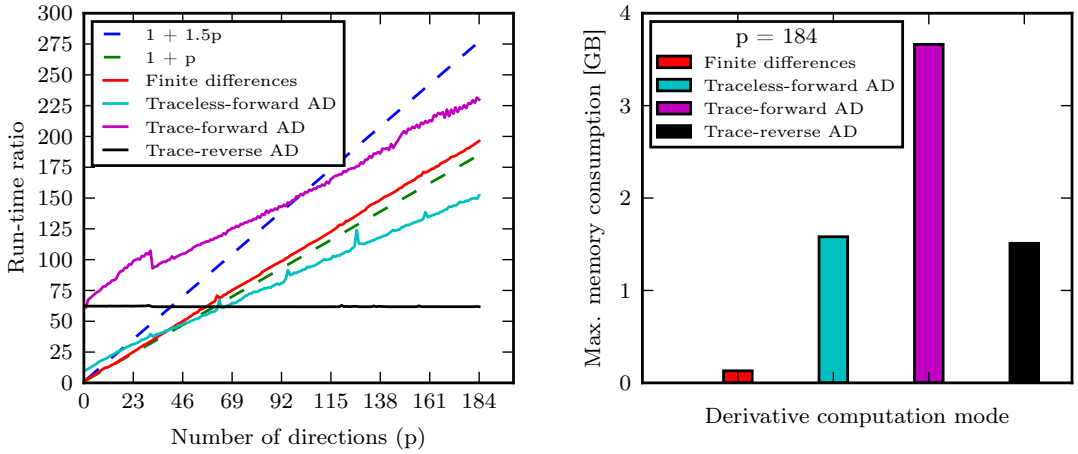


Figure 5.20: Overview of run-time ratios (left) and total memory requirements (right) for the compressor stator blade test case. Figure provided by M. Banovic.

larly when defining many directions (i.e. many design parameters).

5.10 Limitation of the proposed parametrisation approach

As explained in Sec. 3.1 and Sec. 3.2, AD is the most effective technique to compute the derivatives. This technique allows to compute the derivatives exactly and efficiently, if the reverse mode variant of AD is applied. This can be verified in Subsec. 5.6.1 and Subsec. 5.9.1, where it is demonstrated that the reversely differentiated CAD models allow to compute the exact derivatives with the best run-time ratio. The main limitation of the parametrisation approach proposed in this thesis is that such differentiated CAD models are developed by hand, i.e. by coding thousands of C++ lines (Appendix B provides a simplified version of the source code). This approach does not suit the typical workflow of the designer, who is used to work with a Graphical User Interface (GUI) to define the parameters which control the shape (Fig. 5.21).
Sec. 8.4 explains further research work which allows avoiding such limitation (i.e. differentiate the overall CAD system, Fig. 5.21).

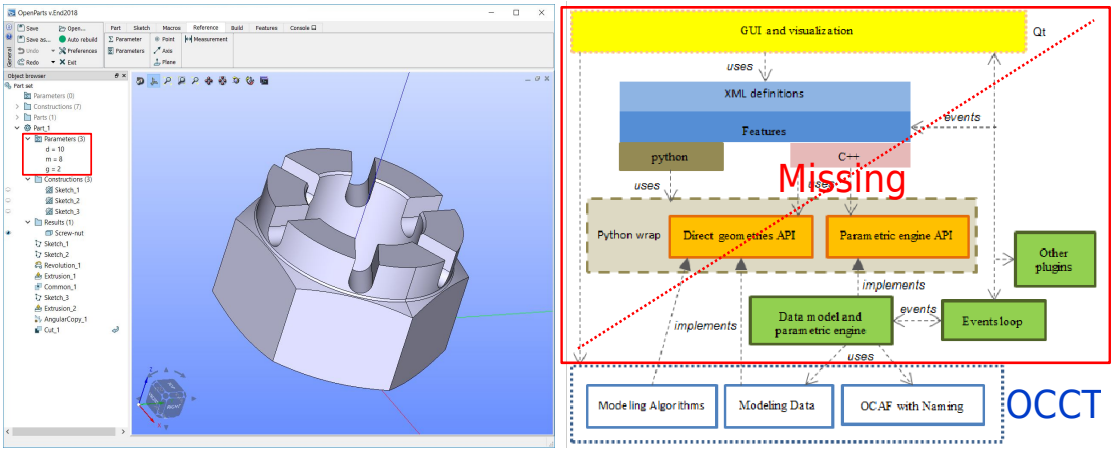


Figure 5.21: On the left, the GUI of a CAD system (Shaper) with the CAD model of the screw nut explained in Sec. 2.3. On the right, the overall structure of this CAD system, with highlighted in the red box the layers of the CAD system which are missing from the current investigation.

5.11 Summary

This chapter has introduced the reader to the two test cases studied in this thesis: a cooling channel for high-pressure turbine blades and a compressor stator blade. The parametrisation of both test cases is based on a cross-sectional design approach. By using this approach, it is possible to compute a free-form surface that approximate a set of 2D sections. The main advantage of this design approach w.r.t. the other approaches provided by **OCCT** is that the shape is locally controlled such that it can be modified based on the particular flow conditions.

The 2D section of the geometry of both test cases is controlled by intuitive parameters. These parameters are implemented by using Bezier/B-spline curves. For the U-bend, each 2D section consists of 4 Bezier curves. The parametrisation of the 2D section of the TUB, which has been introduced in Sec. 4.2, implements the typical airfoil design parameters such as camberline and thickness distribution by employing only B-spline curves. This choice allows to impose that the G2 continuity is kept constant along the 2D section.

The shape derivatives of the CAD models computed with the differentiated OCCT in both traceless forward mode and trace mode are compared against the FD. The AD *trace-reverse* mode requires higher memory consumption then the FD approach, but it implies a lower run-time ratio particularly when defining many directions (i.e. many design parameters).

Chapter 6

Aerodynamic shape optimisation of parametric CAD models

This Chapter presents the design chain used to perform the shape optimisation. The main components of this chain are: the CAD models that parametrise the geometry of the test cases (the U-bend, Sec. 5.2 and the compressor stator blade, Sec. 5.7), the flow solver which evaluates the aerodynamic performance and the optimiser that identifies the optimal parameters.
First, the equations used to efficiently compute the flow derivatives (the *adjoint CFD method*) are introduced in Sec. 6.1. Thereafter, the CAD-based shape optimisation loop is explained in Sec. 6.2. Sec. 6.3 and Sec. 6.4 present the optimisation settings used for the U-bend test case and the flow field of the initial geometry of this test case, respectively. Sec. 6.5 presents the aerodynamic shape optimisation of the U-bend test case. Sec. 6.7 compares the optimisation results obtained by using the parametrisation proposed in this thesis with the NSPCC results. Sec. 6.8 provides the optimisation settings used to perform the shape optimisation of the TUB test case. Sec. 6.9 shows the flow field provided by the initial geometry. Sec. 6.10 demonstrates that the design chain can be successfully applied to perform the flow optimisation of the compressor stator blade.

6.1 Primal and adjoint flow equations

The first way of deriving the adjoint equations in aerodynamic shape optimisation was determined by Jameson [29] and Pirennau [125] by introducing a Lagrangian multiplier argument. Despite this being the first approach, the author prefers to derive the adjoint equations by using the intuitive *linear algebra approach*. The other approaches could be found here [126].

To optimise a scalar cost function J which describes the aerodynamic performance of the system of interest, the optimisation problem can be stated as in Eq. 6.1/Eq. 6.2:

$$\min_{\alpha} \; J(U(\alpha), \alpha), \tag{6.1}$$

subject to (s.t.)

$$R(U(\alpha), \alpha) = 0. \tag{6.2}$$

Eq. 6.2 denotes the system of steady-state Reynolds-Averaged Navier-Stokes equations, where the spatial residual R is driven to zero. R is a function of the non-linear state (or "primal") U and of the design parameters α. The objective function J could correspond to drag, lift, total pressure losses, etc. The dependence of J, R and U on the calculation grid X is hidden to simplify the explanation.
The linearisation w.r.t. α of the Eq. 6.2 provides Eq. 6.3:

$$\frac{\partial R}{\partial U}\frac{dU}{d\alpha} = -\frac{\partial R}{\partial \alpha} \tag{6.3}$$

$$Au = f_p \tag{6.4}$$

In Eq. 6.4, the source term f_p is the negative partial derivative of R w.r.t. α ($f_p = -\partial R/\partial\alpha$). On the left side, there are the Jacobian A and the state perturbation field $u = dU/d\alpha$). The derivative of the objective function J w.r.t. the design variable is expressed in Eq. 6.5:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + \frac{\partial J}{\partial U}\frac{dU}{d\alpha} = \frac{\partial J}{\partial \alpha} + g^T u \tag{6.5}$$

The evaluation of $\frac{\partial J}{\partial \alpha}$ is computationally cheap because J directly depends on α. On the other hand, the calculation of the term $g^T u$ requires, per every design parameter α_i, the evaluation of the perturbation flow field u, which is computationally expensive. It is possible to demonstrate that $\frac{dJ}{d\alpha}$ can be calculated independently from u. Based on Eq. 6.4, one can replace the term u in Eq. 6.5 with the expression $A^{-1}f$:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + g^T A^{-1} f_p \tag{6.6}$$

by transposing Eq. 6.6, we obtain Eq. 6.7:

$$\begin{aligned}\left(\frac{dJ}{d\alpha}\right)^T =& \left(\frac{\partial J}{\partial \alpha} + g^T A^{-1} f\right)^T, \\ & \left(\frac{\partial J}{\partial \alpha}\right)^T + f_p^T A^{-T} g, \\ & \left(\frac{\partial J}{\partial \alpha}\right)^T + f_p^T v\end{aligned} \tag{6.7}$$

where v is the adjoint variable defined by the system of adjoint equations 6.8:

$$\begin{cases} A^T v = g \\ \left(\frac{\partial R}{\partial U}\right)^T v = \left(\frac{\partial J}{\partial U}\right)^T \end{cases} \tag{6.8}$$

The *discrete adjoint relationship* 6.9:

$$g^T u = (A^T v)^T u = v^T A u = v^T f_p, \tag{6.9}$$

derives from the aforementioned demonstration. By transposing back Eq. 6.7:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + v^T f_p \tag{6.10}$$

one retrieves Eq. 6.10 (i.e. the adjoint approach), which can be easily compared to Eq. 6.5 (i.e. the tangent linear approach). Basically, these two equations differ from each other only in the term $g^T u$, which is replaced by $v^T f_p$. The calculation of v^T depends only on the Jacobian A^T, which can be calculated once for all the n design parameters. Moreover, given m cost functions, the term v^T has to be computed m times (Eq. 6.8) because g is function of the objective J (Eq. 6.5).
On the other hand, the calculation costs of the term $g^T u$ scales with the number of design parameters n and is independent on the number of cost functions m. The m cost functions generate a set of m values of g, which only affects the inner product $g^T u$.
Finally, one can state that, for an optimisation problem which considers n design parameters and m cost functions with $n = m$, the two equations 6.5 and 6.10 have similar computational costs. In aerodynamic shape optimisation, $m \ll n$. Therefore, the adjoint approach is much more convenient w.r.t. tangent linear one.

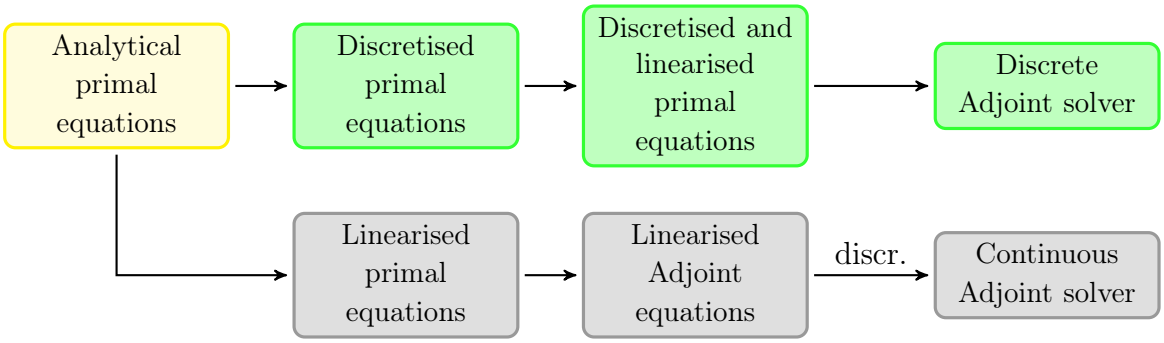


Figure 6.1: Continuous and descrete adjoint method.

6.1.1 The AD discrete adjoint approach

In this subsection, the author focuses on the possible ways the adjoint equations can be derived and discretised. As shown in Fig. 6.1, two approaches can be considered: the continuous and the discrete one. The continuous adjoint equations [29, 125] are usually derived using the Lagrangian multiplier. This discretisation approach first linearises the primal equations and derives the adjoint equations. Then, these equations are discretised.
The discrete approach [127–129] first discretises the flow equations 6.2. Then, the instructions of the discretised algorithm are differentiated in order to obtain the adjoint code. The main advantage of the discrete approach is that it allows to rigorously verify blocks of explicit code of adjoint against tangent linear. These two codes must provide the same derivative values (i.e. below machine precision). However, the comparison between the two codes (tangent linear and adjoint) does not guarantee that these codes present no errors. It could happen that both of them are provided with the same error. Therefore, a comparison between the derivative values given by AD and the ones computed with finite differences is necessary.
The discrete adjoint method can be implemented or manually (i.e. hand-differentiation) or by using AD. Hand-differentiation could be tedious and error-prone and, thus, the AD implementation has gained several followers [130–133]. The fundamental principles of AD are presented in Sec. 3.2.

6.1.2 Adjoint CFD with AD: STAMPS solver

CFD solver: STAMPS

STAMPS (Source Transformation Adjoint Multi-Purpose Solver [134]) is a CFD/Adjoint CFD solver developed by the group of CFD optimisation at Queen Mary

University of London. It has been retrieved by adding to the solver Mgopt [135] new features for turbomachinery applications and for performing calculations with parallel computing architectures (interface used: MPI). It is developed in `Fortran 90` in order to allow the application of the AD tool Tapenade [94] to its sources. STAMPS is a vertex-centred finite volume solver with edge-based fluxes and is developed for unstructured grids. MUSCL is the scheme used to retrieve the spatial second order convective term. An edge-corrected Green-Gauss formula [136,137] is used to compute the viscous source terms. The turbulence model implemented is the Spalart-Allmaras [138] (RANS model). References [37,135] give details about numerical method and implementation.

Adjoint CFD in STAMPS

The Adjoint system in Eq. 6.8 is obtained by applying the AD technique (reverse mode, Sec. 3.2) to the primal steady-state problem in Eq. 6.2. The reverse mode of AD stores in memory (or in a tape) the variables that are needed for the computation of the derivatives. STAMPS (such as other Adjoint AD tools) identifies which variables are needed and which variables are not necessary to provide such computation. The latter variables are finally omitted. Moreover, the application of the AD tool TAPENADE [94] to the primal has been careful in order to prevent the differentiated code from having unaffordable memory requirements. These requirements have been consistently reduced by pre-processing and restructuring the primal [130,139]. The differentiation of the steady-state pseudo-timestepping CFD code implemented in STAMPS is achieved by submitting the spatial discretisation to TAPENADE such that the differentiated code consists of subroutines reassembled in the fixed-point time-stepping scheme [139].

6.1.3 Mesh Deformation Algorithm

The comparison between the tangent linear approach (Eq. 6.5) and the adjoint one (Eq. 6.10) has been conducted in Sec. 6.1 by hiding the dependence of J, R and U on the computational grid X.
These dependencies are needed to explain the role of the mesh deformation algorithm in the computational chain used in this thesis. Given the cost function $J(U(X(\alpha), X(\alpha), \alpha))$ subject to $R(U(X(\alpha)), X(\alpha))$, one can retrieve Eq. 6.11 from

the Eq. 6.10:

$$\frac{dJ}{d\alpha} = \left[\frac{\partial J}{\partial X} + v^T f\right] \frac{\partial X}{\partial X_S} \frac{\partial X_S}{\partial \alpha} \tag{6.11}$$

where X_S represents the surface mesh points and $f = -\partial R/\partial X$. One can transpose Eq. 6.11 in order to compute the total derivative $\frac{dJ}{d\alpha}$ with the efficient adjoint method, Eq. 6.12:

$$\frac{dJ}{d\alpha}^T = \frac{\partial X_S}{\partial \alpha}^T \underbrace{\frac{\partial X}{\partial X_S}^T \left[\frac{\partial J}{\partial X} + v^T f\right]^T}_{\text{CFD derivative}} \tag{6.12}$$

In Eq. 6.12, we can identify two terms: the CFD derivative shown in Eq. 6.13 and the shape derivative $\frac{\partial X_S}{\partial \alpha}$.

$$\frac{dJ}{dX_S}^T = \frac{\partial X}{\partial X_S}^T \left[\frac{\partial J}{\partial X} + v^T f\right]^T \tag{6.13}$$

In Eq. 6.13, the term in brackets is computed by the flow solver differentiated in reverse mode such that the CFD derivative is finally calculated by right-multiplying $\frac{\partial X}{\partial X_S}$. $\frac{\partial X}{\partial X_S}$ is provided by the mesh deformation algorithm.

At each iteration of the optimisation, the modified optPart (i.e. the part of the CAD model that has to be optimised) requires a deformation of the mesh. This deformation is possible using a mesh deformation algorithm. This is an important part of the optimisation loop because the mesh must keep a sufficient quality (which can be evaluated by checking the mesh quality factors such as skewness or aspect ratio) to guarantee the stability of the solver.

Several mesh deformation algorithms have been developed for unstructured meshes such as analytic approach [140], the spring-analogy [141], the radial basis function (RBF) [142] and the inverse distance weighting (IDW) approach [143].

The analytic approach (e.g. FFD) traces the volume mesh onto an object for which the analytic representation is available. This object is smoothly perturbed and then this perturbation is reported onto the volume mesh nodes. The spring-analogy defines the mesh as a spring network. Basically, every node of the mesh is seen as connected to the others nodes with a spring whose stiffness is inversely proportional to the length of the edge that connects the two nodes. Radial basis function (RBF) approach evaluates the volume mesh deformation with a linear combination of various radial basis functions. Per every function, the coefficient is calculated based on the displacement of the surface nodes. The main benefit of such approach is its robustness [142] whereas the principal disadvantage is that it

is computationally expensive [142].
Inverse distance weighting (IDW) [143] is an explicit method that computes the inner mesh point displacements based on the displacements of all the surface mesh points. Its mathematical formulation is presented in Eq. 6.14:

$$\delta X_i = \frac{\sum_{j \epsilon \partial\Omega} W(||X_i - X_S^j||)\delta X_S^j}{\sum_{j \epsilon \partial\Omega} W(||X_i - X_S^j||)} = \sum_{j \epsilon \partial\Omega} d_i^j \delta X_S^j = D\delta X_S \quad i\epsilon\Omega. \tag{6.14}$$

where W represents the weighting function related to inverse-distance $||X_i - X_S^j||$ and Ω and $\delta\Omega$ are the volumetric and surface domains. The differentiation of Eq. 6.14 provides Eq. 6.15:

$$\frac{\partial X}{\partial X_S}^T = [d_i^j]_{mxm_b}^T = D^T. \tag{6.15}$$

In Eq. 6.15, m is the size of the volume mesh and m_b represents the size of the surface mesh.
The IDW approach does not rely on the mesh connectivity information. Wittewen [143] demonstrated that the IDW interpolation guarantees a mesh quality similar to the one provided by RBF. Moreover, being an explicit method, it allows to retrieve the surface derivatives projections only for the surface mesh points of the optPart (i.e. the part of the CAD model to be optimised). This reduces the computational cost of this approach. IDW is the approach utilised in this research.

6.2 CAD-based shape optimisation

Among the parameters controlling the CAD model (Subsec. 2.3.1), we can identify a set of them (α_i with $i = 1 \dots n$: n is the number of design parameters) that governs the optPart, i.e. the part of the CAD model that has to be aerodynamically optimised (e.g. the U-turn of the U-bend, Sec. 5.5). In a CAD-based design chain, the set of parameters α represents the design parameters of the optimisation.
By using a CAD-based design chain, the terms of Eq. 6.12 can be therefore split into: (i) the CFD derivative $\frac{dJ}{dX_S}$ (Eq. 6.13) and the CAD derivative $\frac{\partial X_S}{\partial \alpha}$, which is the derivative of the surface mesh points of the optPart w.r.t. the set of parameters α. The first term calculates the change in the cost function due to the infinitesimal movement of the surface mesh points. The second term computes the perturbation

of the surface mesh points related to a modification of the design parameters. The assembling of these two terms provides the total derivative $\frac{dJ}{d\alpha}$ shown in Eq. 6.12. This total derivative represents the derivative of the cost function J w.r.t. the design parameters of the optPart of the CAD model.
The assembling between CFD and CAD derivative can be done at surface mesh points level. Given the set of surface mesh points q, the CFD derivatives in the surface mesh points are expressed in Eq. 6.16:

$$\frac{dJ}{dX_S} = \left[\frac{dJ}{dX_{S,i}}\right] \quad i = 1 \dots 3q \tag{6.16}$$

whereas the CAD derivatives can be defined as in Eq. 6.17:

$$\frac{dX_S}{d\alpha} = \left[\frac{dX_{S,i}}{d\alpha_j}\right] \quad j = 1 \dots n \tag{6.17}$$

The size of the derivatives shown in Eq. 6.16 and Eq. 6.17 is $3xq$, if we are considering 3D optimisation problems. After having computed all the derivative shown in Eq. 6.16 and Eq. 6.17, we can assemble them as in Eq. 6.18:

$$\frac{dJ}{d\alpha_j} = \sum_{i=1}^{3q} \frac{dJ}{dX_{S,i}} \frac{dX_{S,i}}{d\alpha_j} \tag{6.18}$$

6.2.1 Fully differentiated design chain

The reverse mode implementation of OCCT allows to plug as seed the CFD derivatives $\frac{dJ}{dX_S}$:

$$\frac{dJ}{d\alpha}^T = \frac{\partial X_S}{\partial \alpha}^T \underbrace{\frac{\partial J}{\partial X_S}^T}_{\text{seed}} \tag{6.19}$$

such that a fully differentiated design chain is finally defined. In this thesis, the terms necessary to compute the total derivative $\frac{dJ}{d\alpha}$ (Eq. 6.12) are computed as follows:

- $\left[\frac{\partial J}{\partial X} + v^T f\right]^T$ is computed by the adjoint CFD solver STAMPS (Subsec. 6.1.2).
- $\frac{\partial X}{X_S}^T$ is computed with the mesh deformation algorithm automatically differentiated in reverse mode.
- The CAD derivative $\frac{\partial X_S}{\partial \alpha}^T$ is computed by the differentiated OCCT in reverse mode.

6.2.2 Optimisation framework

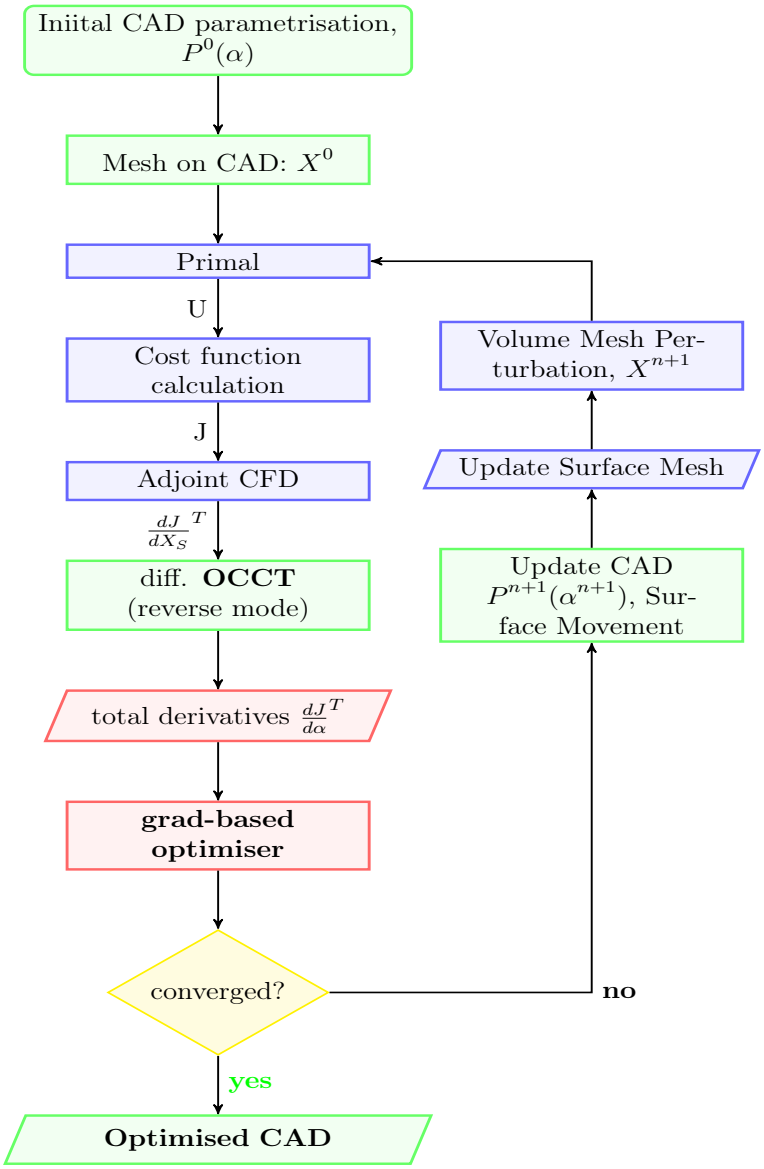


Figure 6.2: Optimisation framework using the differentiated **OCCT** in RM. The differentiated CAD kernel is responsible for the calculation displayed in the green ellipses. *STAMPS* (CFD/Adjoint CFD solver) performs the computations shown in the blue ellipses.

The steps of CAD-based optimisation workflow are shown in Fig. 6.2. The design parameters α are used to construct the CAD model (Subsec. 5.5.3 and Subsec. 5.8.2). The CAD model is meshed and, for the first iteration of the optimisation, the design mesh points (i.e. the surface mesh points of the part of the CAD model that has to be optimised) are projected onto the optPart. The (u, v) values of these design mesh points are stored. The execution of the primal and the

adjoint solver allows to compute the CFD derivatives $\frac{dJ}{dX_S}^T$ on the current grid. These CFD derivatives are plugged as seed vector into OCCT differentiated in reverse mode (Sec. 3.4), which calculates the total derivatives (Eq. 6.12). The total derivatives are finally given to a gradient-based optimiser, which provides the set of design parameters that generate the CAD model with improved performance (i.e. improved cost function). These improved design parameters are used to generate the new CAD model. The design mesh points of the optPart are retrieved by using the (u, v) of values of the design mesh points stored from the previous optimisation iteration. Based on the displacement of the design mesh points, the computational grid is updated by the mesh deformation algorithm (Subsec. 6.1.3) and another optimisation iteration is performed. Finally, the simulation stops when the optimisation has converged.

6.3 U-bend Optimisation: settings

6.3.1 Simulation settings

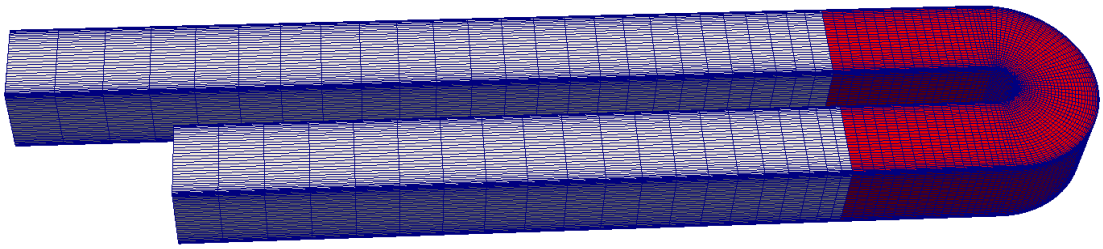


Figure 6.3: U-bend computational grid. In red, the computational grid of the optPart.

In this section, the settings used to perform the CAD-based shape optimisation of the U-bend test case (described in Sec. 5.5) are presented. This this test case has been identified within the EC project IODA[1]. In this section, the author reports the main settings used to perform the aerodynamic shape optimisation [9, 144]. The objective function to minimise is the total pressure losses between the inlet and the outlet channel, which is defined as in Eq. 6.20:

$$J = \frac{\int_{inlet} p_{total}(u_v \cdot n)\, dS - \int_{outlet} p_{total}(u_v \cdot n)\, dS}{\int_{inlet}(u_v \cdot n)\, dS} \tag{6.20}$$

[1] `https://ioda.sems.qmul.ac.uk/`

where p_{total} is the total pressure, u_v is the velocity vector, n is the normal direction and S is the cross-section area of the channel. The Reynolds number is 15000.
The CFD computations are performed by the in-house CFD/adjoint CFD solver *STAMPS* [145]. As explained in Subsec. 6.1.2, STAMPS is a compressible flow solver. The flow is modeled as incompressible by fixing the Mach number at 0.1. The turbulence model used is the Spalart-Allmaras (SA), that is also the unique model currently implemented in *STAMPS*. Despite this turbulence model being mainly used for external flows, it can be successfully used also for the U-bend test case, as demonstrated by Verstraete et al. [120]. The computational grid (the regular hexahedral mesh shown in Fig. 6.3) consists of 167K nodes and 177K cells. This mesh is refined at the boundary layer such that the secondary flow can be evaluated with a low Reynolds turbulence model. The average y^+ value is 1.
Other researcher at QMUL [146] have been conducting a mesh convergence study by defining three mesh levels (coarse, medium and fine), with the medium and the fine mesh which consist of 167k nodes and 300k nodes, respectively. The results of the mesh convergence study demonstrate that the difference in terms of the cost function value between the medium and the fine mesh is just 0.3%. The mesh with 167k nodes has therefore been chosen for this test case.
The test case imposes the following boundary conditions:

- inlet: flow velocity ($U_0 = 8.4\ m/s$), turbulent intensity (5%) and flow temperature (293.15 K).
- Outlet: constant pressure (1 bar).

Further details about this test case have been specified on the About Flow website[2], which is the IODA parent project within the framework of the Marie Curie action.
The optimisation loop is the one shown in Fig. 6.2, where the CAD model is constructed by using the feature tree explained in Sec. 5.5. The optimiser is the steepest descent (SD) method. This method is defined as in Eq. 6.21:

$$\alpha_{k+1} = \alpha_k - s_k d(\alpha_k) \tag{6.21}$$

where α_{k+1} is the design variable at the iteration $k+1$, s_k is the step size and $d(\alpha_k)$ is the gradient of the cost function at the point α_k.
This method can be easily implemented because it does not utilise the history information of the gradient to compute the new set of design variables x_{k+1}. On

[2] `https://aboutflow.sems.qmul.ac.uk/events/munich2016/benchmark/`

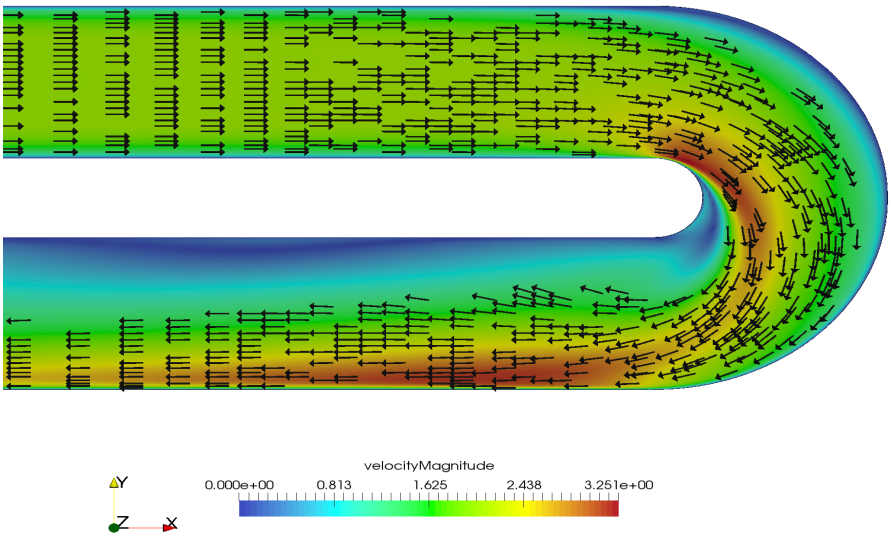


Figure 6.4: U-bend: initial velocity magnitude at mid height plane.

the other hand, this first order method is not really efficient and could have poor performance (slow convergence) if used for complex simulations. Despite the poor performance, the explicit control of the design steps made parametrisation updates and corresponding volume mesh movement robust. For this reason, SD was finally chosen as optimisation method.

6.4 U-bend: initial flow field

Fig. 6.4 shows the initial flow field of the U-bend at mid height plane. After the U-turn, there is a large flow separation that continues in the downstream leg. The strong curvature of the U-turn implies an acceleration of the flow at the inner wall and a deceleration close to the outer wall. The high velocity of the flow close to the inner wall causes a reduction of the pressure, which brings to an adverse pressure gradient in the downstream leg. Fig. 6.5 shows the initial velocity magnitude at several locations of the initial geometry. After the U-turn, the flow separation reduces the effective cross-section area. As a consequence, the flow close to the outer wall is accelerated. This flow distribution evolves in the downstream leg such that the flow reattachment to the inner wall is very slow. This implies a reduction of the total pressure of the flow between the inlet and the outlet of the channel. Finally, as shown in Fig. 6.5, two counter-rotating vortices ("Dean" vortices) are present at 90° position of the U-part. The Dean vortices are generated by the imbalance between the centrifugal force and the pressure gradient, which is due to the reduced velocity near the endwalls. The Dean vortices affect the evolution

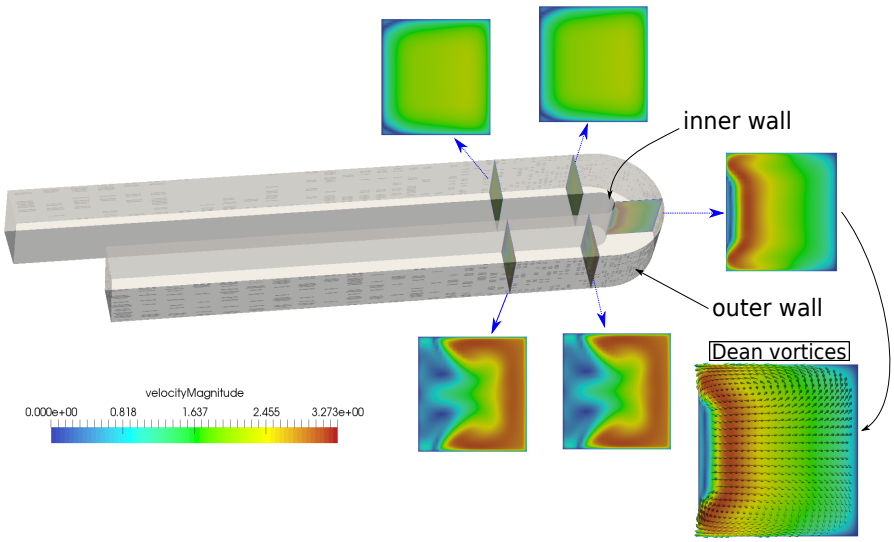


Figure 6.5: U-bend: initial velocity magnitude at different locations.

of the flow in the outlet leg of the U-bend.
The main goal of the shape optimisation is to determine the shape of the U-turn which does not cause the vortex generation or which implies a limited vortex that rapidly reattaches to the inner wall. This would allow to reduce the pressure losses between the inlet and the outlet of the bend.

6.4.1 Comparison with experimental results

Coletti et al. [147] have analysed the U-bend baseline geometry with experimental tests. A simplified scheme of the testbed settings used for this analysis is shown in Fig. 6.6. In this scheme, particle image velocimetry (PIV) measurements are used to evaluate the velocity flow field.
The comparison of the velocity magnitude at mid height plane of the baseline geometry between numerical analysis (solver used: STAMPS) and experimental tests is shown in Fig. 6.7. The flow calculated by STAMPS is very close to the one provided by the experimental results. The main flow conditions verified by both numerical calculation and experimental test are as follows: (i) the flow accelerates close to the inner wall, (ii) the flow presents a strong detachment after the U-turn. This qualitative comparison confirms that the turbulence model chosen (the S-A) is able to reproduce the main features of the flow field. The goal of the shape optimisation is to reduce the total pressure loss between the inlet and the outlet of the channel (Eq. 6.20).

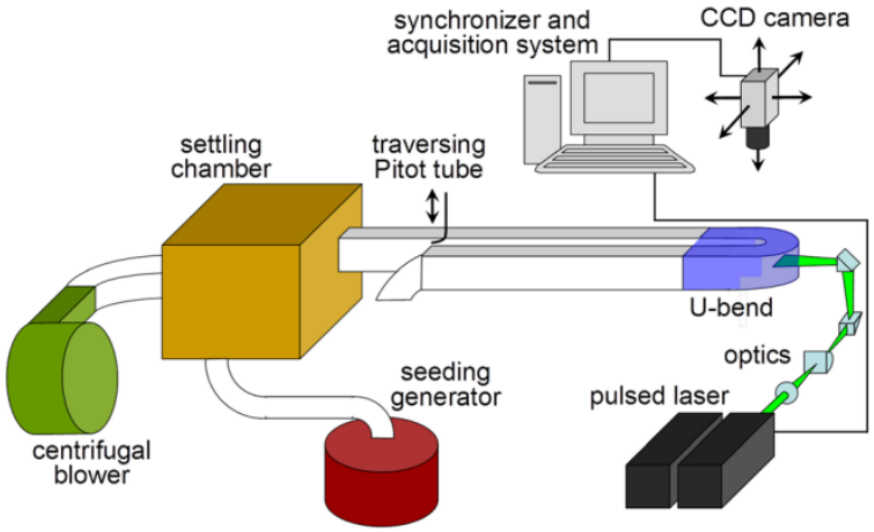


Figure 6.6: U-bend: scheme of the experimental set up used to test the initial geometry of the U-bend test case. Full explanation of the test can be found here [147].

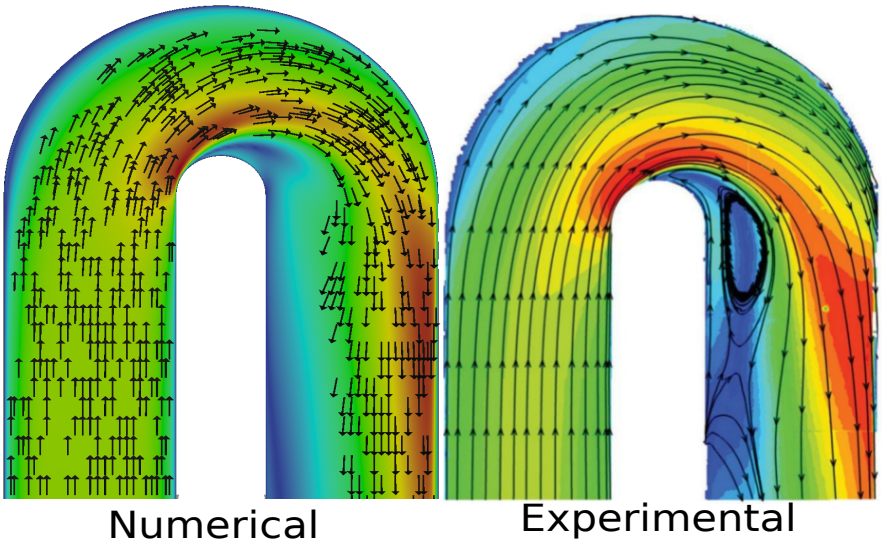


Figure 6.7: U-bend: comparison between the numerical results provided by STAMPS and the experimental velocity magnitude presented by Coletti et al. [147] at mid height plane of the baseline geometry.

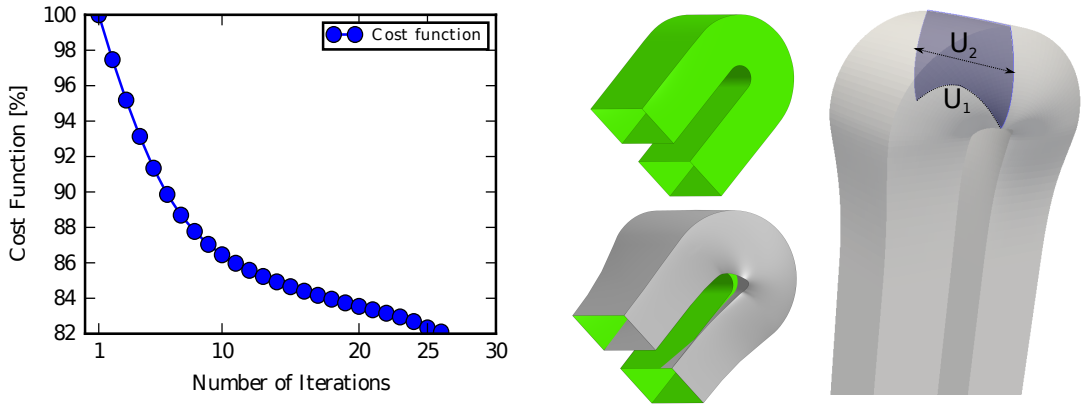


Figure 6.8: Cost Function trend vs number of iterations (left) and initial (green) and optimised (gray) CAD model (right).

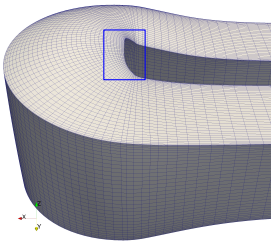


Figure 6.9: U-bend mesh at iteration 26, with the distorted area at the U-turn highlighted by the blue box.

6.5 U-bend: optimisation results

The optimisation history is shown in Fig. 6.8. After 26 iterations, the optimised shape is able to reduce the ΔP between the inlet and the outlet of the U-bend by 18%. The optimisation is not fully completed because the quality of the deformed mesh became too poor for the flow solver to converge (Fig. 6.9). As the design chain provides over an exact CAD description of the geometry, one could remesh and run further optimisation steps. Remeshing is currently a manual step which is not included in the automated design algorithm. In order to complete the shape optimisation, it would be necessary to implement and test other mesh deformation algorithms (Subsec. 6.1.3). This is one of the next steps of this research.

However, the parametric-based results shown in this thesis are useful to give a first application of the fully differentiated design chain and to verify if the parametrisation (Sec. 5.5) is able to reproduce an optimal shape with the geometric features similar to the ones given by the reference results (Sec. 6.6).

The improvement in the cost function shown in Fig. 6.8 is mainly due to the re-

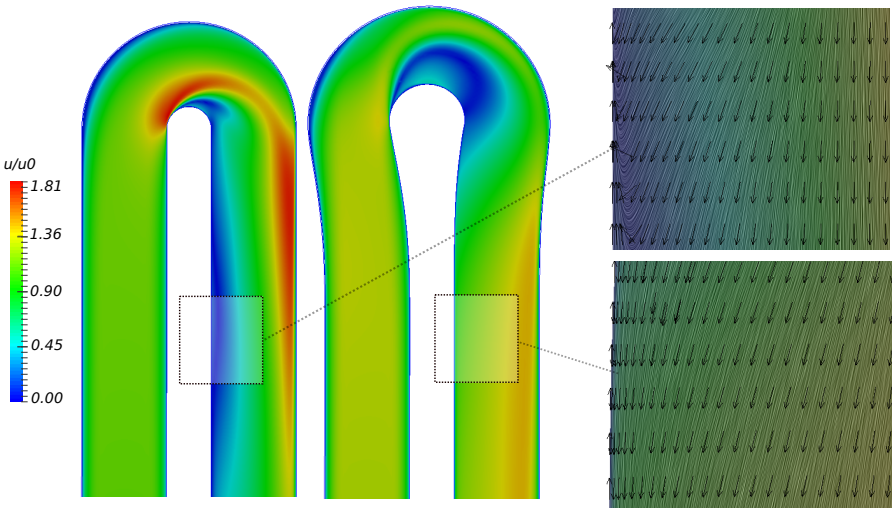


Figure 6.10: Left: baseline and optimised mid-span velocity magnitude. Right: velocity vector in the outlet leg.

duction of the flow separation after the U-turn and downstream in the outlet leg. As shown in Fig. 6.10, the CAD-driven optimisation managed to eliminate the flow separation at the outlet leg, as well as the corresponding reverse flow motion present in the initial design. These aerodynamic improvements are related to the geometrical differences between the initial and the optimised geometry. First, the inner wall of the optimised U-bend has an increased radius of curvature (which in this thesis will be referred as the design mode number U_1, Fig. 6.8). Then, the section of the channel of the U-turn has a bigger diameter (width, design mode number U_2, Fig. 6.8). These two main changes of the geometry affect the centrifugal force (Eq. 6.22) to which the flow is subject:

$$F_{centrifugal} = m \cdot \frac{v^2}{r} \tag{6.22}$$

where: m is the mass, v is the velocity of the flow and r is the radius. A bigger inner U-turn radius of curvature implies that the acceleration of the flow from the inner wall to the outer one is slower.

$$\dot{m} = \rho \cdot A \cdot v \tag{6.23}$$

Moreover, an incremented channel diameter causes a reduction of the flow velocity. Considered constant the mass flow rate passing through the U-turn (Eq. 6.23) and the density ρ, a bigger area A results in a reduction of the flow velocity. This also contributes to decrease the centrifugal force such that the acceleration of the

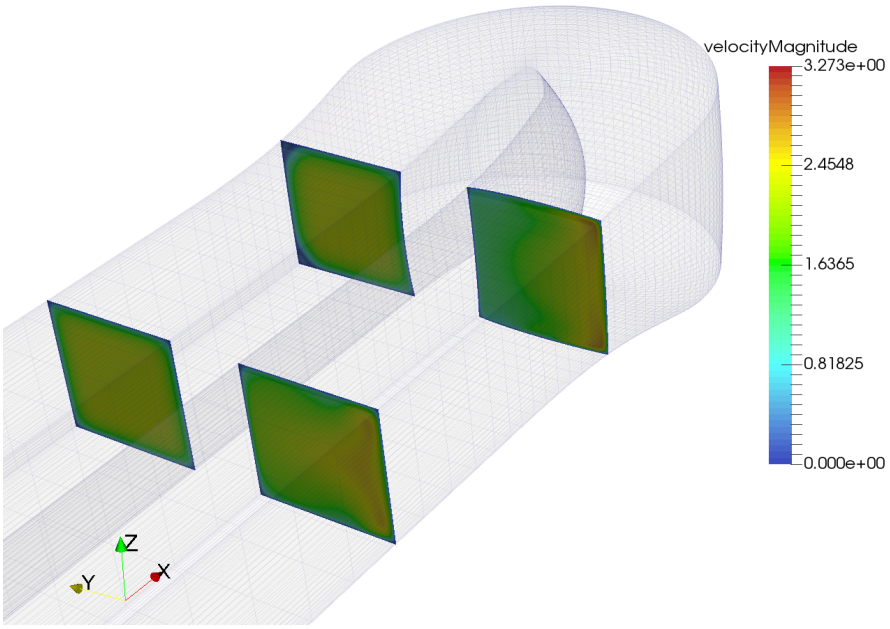


Figure 6.11: Velocity magnitude at different locations of the optimised U-bend.

flow from the inner to the outer wall is further reduced. The reduction in the flow acceleration causes also a decreased radial pressure gradient. The streamwise adverse pressure gradients is limited and the related vortex is reduced. This allows a rapid reattachment of the flow to the inner wall. Fig. 6.11 reports the velocity magnitude at different locations of the optimised U-bend.

As shown in Fig. 6.12, the flow acceleration across the U-part caused by the baseline CAD model is prevented by the optimised geometry. This contributes to reduce the ΔP.

Finally, Fig. 6.13 presents a comparison of the static pressure distribution at mid height plane between the baseline and the optimised geometry. First, one can verify that across the U-turn ("A" in figure) the pressure gradient between the inner and the outer wall is consistently reduced. Then, the low pressure area at the U-turn that generates the flow detachment in the downstream leg is almost nullified ("B" in figure). Finally, the pressure drop between the inlet and the outlet of the optimised bend is decreased w.r.t. the baseline geometry. Such decrease is demonstrated by the evidence that the optimised geometry requires a static pressure at the inlet ("C" in figure) which is reduced w.r.t. the baseline geometry. Fig. 6.13 also shows the comparison between the static pressure distribution at 90° position of the U-part. The optimised geometry provides a very smooth pressure gradient going from the inner wall to the outer one. This is not given by the baseline geometry.

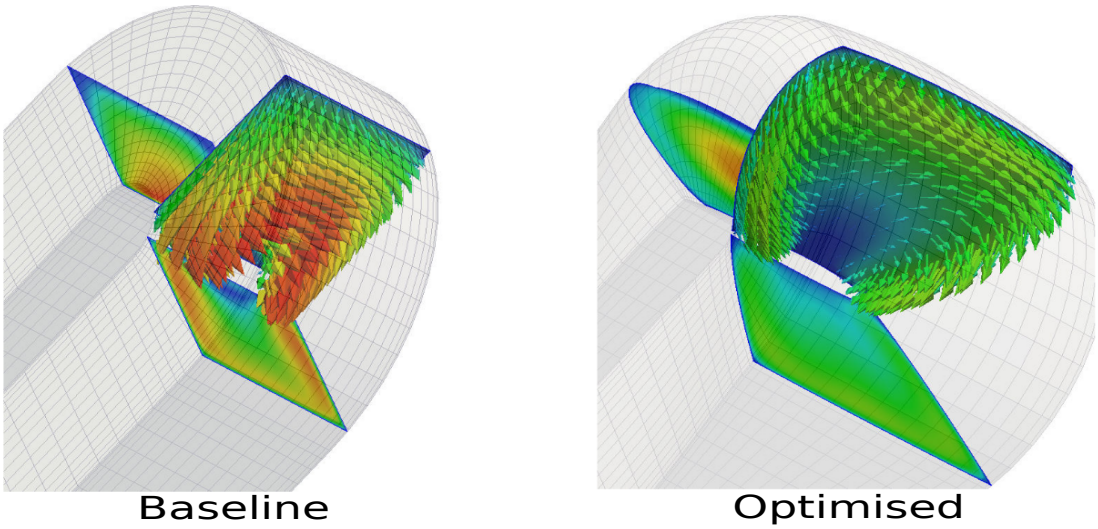


Figure 6.12: U-bend: mid U-turn velocity vectors and static pressure distribution.

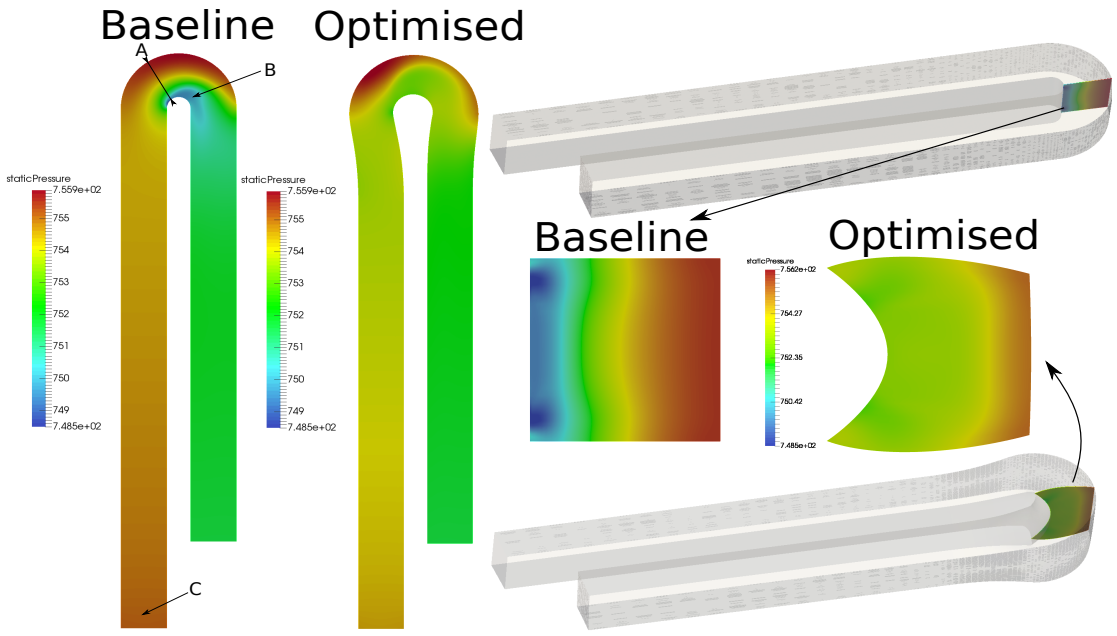


Figure 6.13: U-bend: comparison between static pressure at the mid height plane, on the left. Comparison between the static pressure distribution at at 90° position of the U-part, on the right.

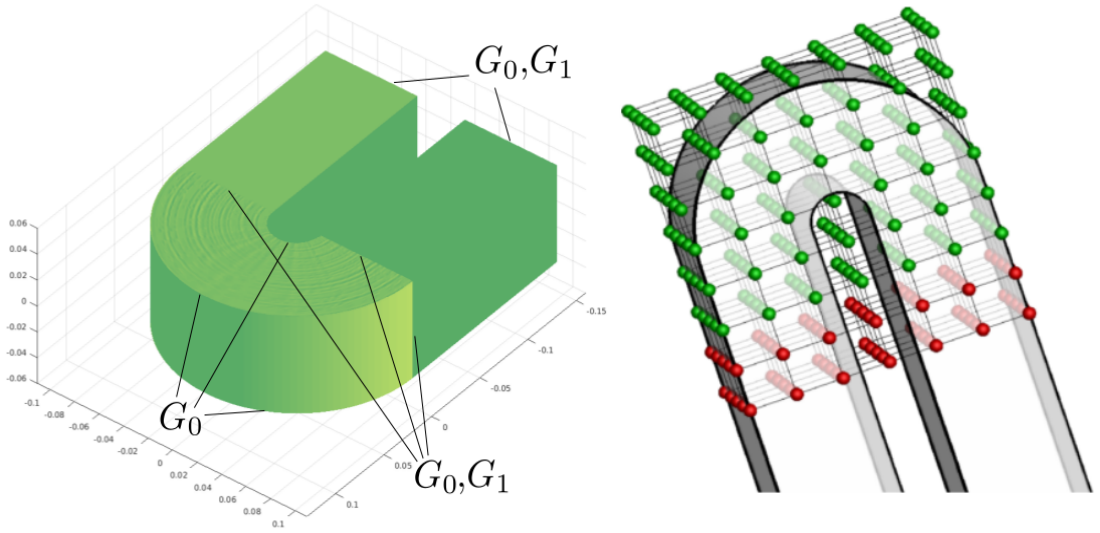


Figure 6.14: U-bend: NSPCC parametrisation, as reported by Zhang [5].

6.6 U-bend: refined CAD-based parametrisations

This section provides the results considered as reference, i.e. the results to be compared to the ones obtained in this thesis. The author identifies as reference parametrisation the one that allows to reduce the cost function the most. Despite the mesh being the most refined design space, node-based parametrisations (Sec. 2.2) are not considered by the author as reference parametrisation. As shown by Jesudasan [148] for bend-like test cases, the regularisation normally required by node-based parametrisations removes some of the high-frequency modes such that the identification of an optimal shape is finally hindered. Jesudasan [148] demonstrates that NSPCC ("NURBS-based parametrisation with complex constraints", Subsec. 2.3.2), which has been implemented by other researchers at CFD optimisation group of QMUL [28,89,90], is the approach which reduces the cost function the most.
This section provides the main results obtained with the NSPCC approach for the U-bend test case [5, 149].

6.6.1 U-bend optimisation with NSPCC: first results

The first author to use the NSPCC approach to parametrise the geometry of the U-bend has been Zhang [5]. By using this approach, he imposes the patch continuity constraint (G0, G1, Fig. 6.14) between the 8 rectangular NURBS and the 4 circular ones that compose the optPart (i.e. the part of the U-bend to be

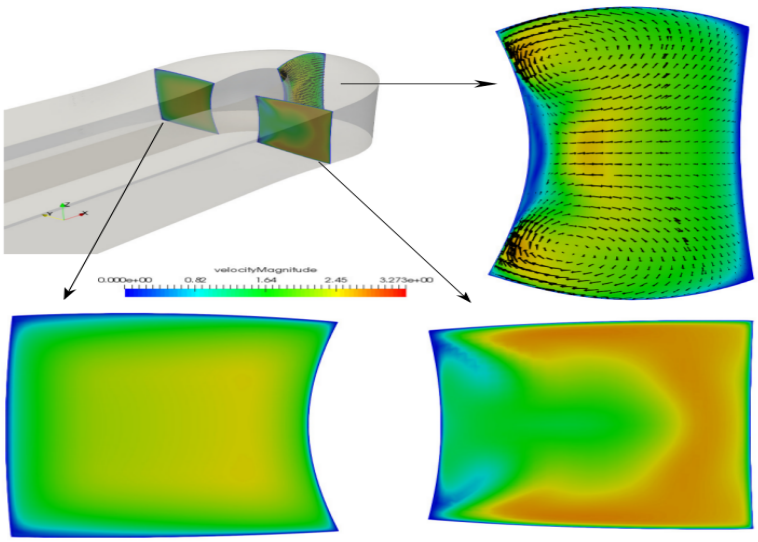


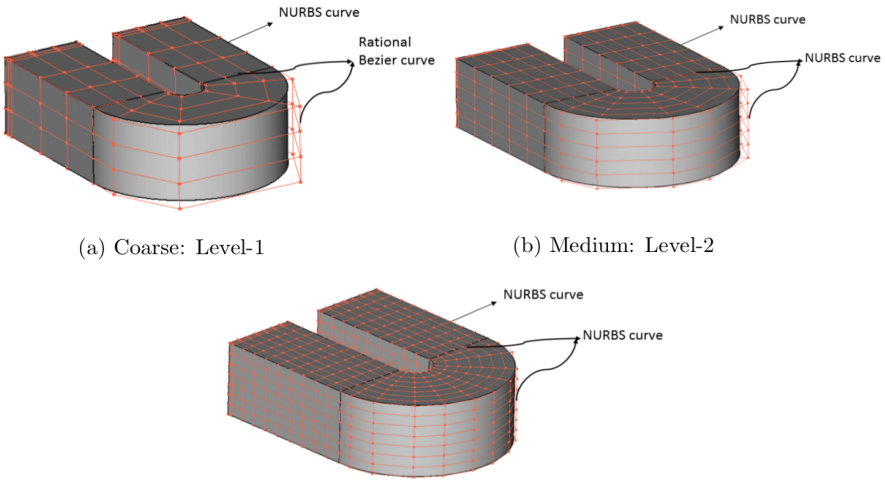
Figure 6.15: U-bend: velocity magnitude at different locations of the shape optimised with the NSPCC parametrisation, as reported by Zhang [5].

optimised, Sec. 5.5) of the geometry. Fig. 6.14 shows the overall set of control points that governs the optPart of the U-bend. This set is used by Zhang to perform the aerodynamic shape optimisation. The control points in red are fix during the flow optimisation in order to assure the G1 continuity between the optPart and the inlet/outlet legs of the U-bend.

The optimisation performed by Zhang converges after 32 iterations. The optimal shape, that is controlled by 445 design variables and that is able to reproduce the two design modes U_1 and U_2 (Sec. 6.5), reduces the cost function by 23.1%. The velocity magnitude at different locations of the optimised geometry is shown in Fig. 6.15.

6.6.2 U-bend optimisation with NSPCC: the latest results

Jesudasan provides three different refinement of the design space. These three parametrisations, which are shown in Fig. 6.16, are defined by 8 rectangular patches and 4 circular ones. Level-1 (L1) parametrisation is governed by 288 control points (each patch has 6×4 control net). Level-2 (L2) considers per every patch a bi-cubic NURBS with 8×6 control net such that the total number of control points is 576. Level-3 (L3) is defined by using bi-cubic NURBS patch with 12×8 control net and the number of control points is 1152. The difference in terms of cost function reduction between the three parametrisations is not huge. As shown in Table 6.1, the parametrisation which reduces the cost

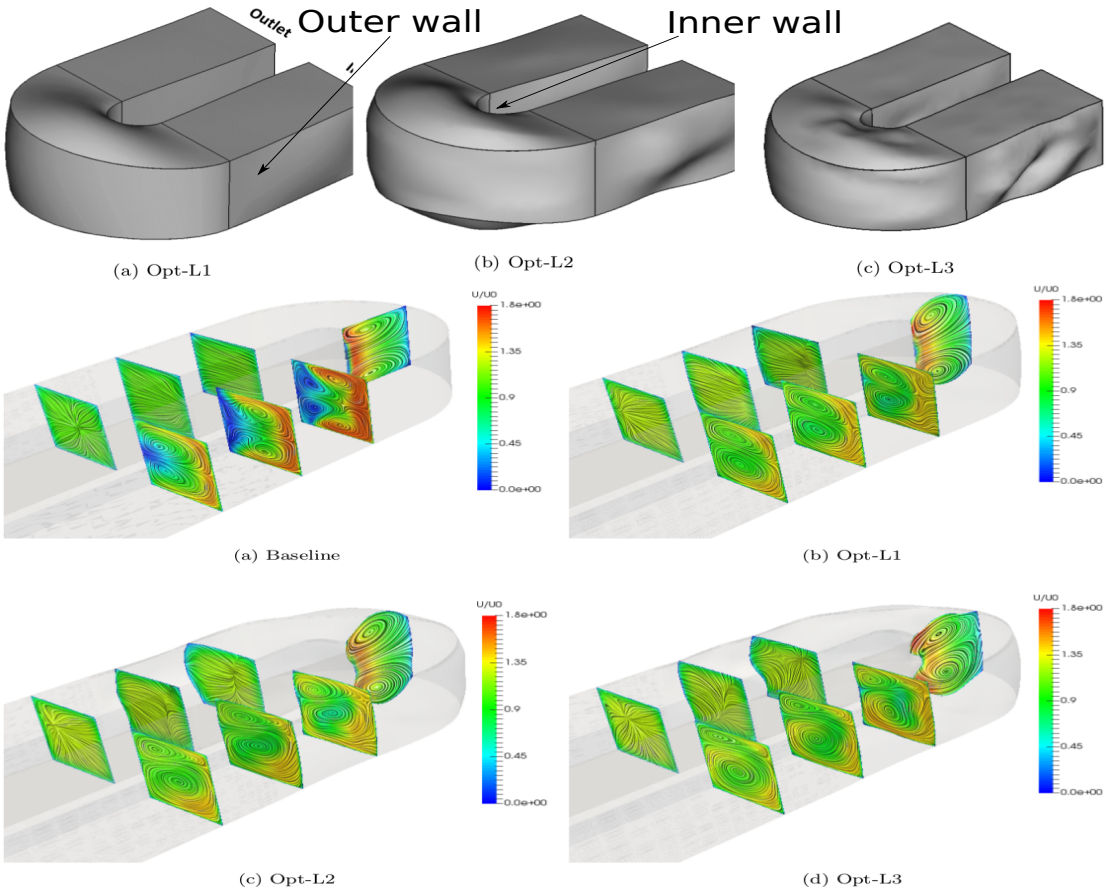


(a) Coarse: Level-1

(b) Medium: Level-2

(c) Fine: Level-3

Figure 6.16: U-bend: NSPCC parametrisation, as reported by Jesudasan [149].



(a) Opt-L1

(b) Opt-L2

(c) Opt-L3

(a) Baseline

(b) Opt-L1

(c) Opt-L2

(d) Opt-L3

Figure 6.17: U-bend: NSPCC results, as reported by Jesudasan et al. [149, 150]

Parametrisations	Total number of control points	% drop in cost function
L1	288	25.34%
L2	576	26.67%
L3	1152	27.52%

Table 6.1: U-bend: NSPCC optimised cost function values, as reported by Jesudasan et al. [149].

function the most (27.52%) is the level-3 (L3), whilst the other two parametrisations provide 26.67% (L2) and 25.34% (L1) reduction. All the three optimised shapes (i.e.Opt-L1, Opt-L2 and Opt-L3) present the two design modes U_1 and U_2 (Sec.6.5): (i) the radius of curvature of the inner wall is increased (U_1) and (ii) the section of the channel of the U-turn is provided with a bigger diameter (width, U_2). The L3 parametrisation also catches surface undulations on the outer wall of the channel (Fig. 6.17) which are not given by the Opt-L1. Opt-L2 slightly reproduce such undulations. This is mainly due to the local control assured by the L3 parametrisation w.r.t. the other two.

6.7 Parametric-based vs NSPCC results

A qualitative comparison between the optimal shape obtained with the parametric-based design (i.e. the parametrisation proposed in this thesis, Sec. 5.5) and the optimal shape provided by the NSPCC approach is presented in this section. This comparison is considered an intermediate step because the optimisation of the parametric-based design has not fully converged. However, this comparison is useful to verify if the parametrisation used in this thesis (Sec. 5.5) is able to reproduce the key geometric features necessary to successfully perform the shape optimisation.
Subsec. 6.7.1 provides the comparison w.r.t. the results given by Zhang [5], whereas Subsec. 6.7.2 gives the comparison w.r.t. the latest results which will be presented by Jesudasan et al. [149].

6.7.1 First NSPCC results

Fig. 6.18 reports the comparison between the flow distribution provided by the optimal shapes obtained with the parametric-based design and the NSPCC parametrisation. A general agreement between the two shapes can be evaluated. Both parametrisations increase the inner radius of the optPart (design mode number

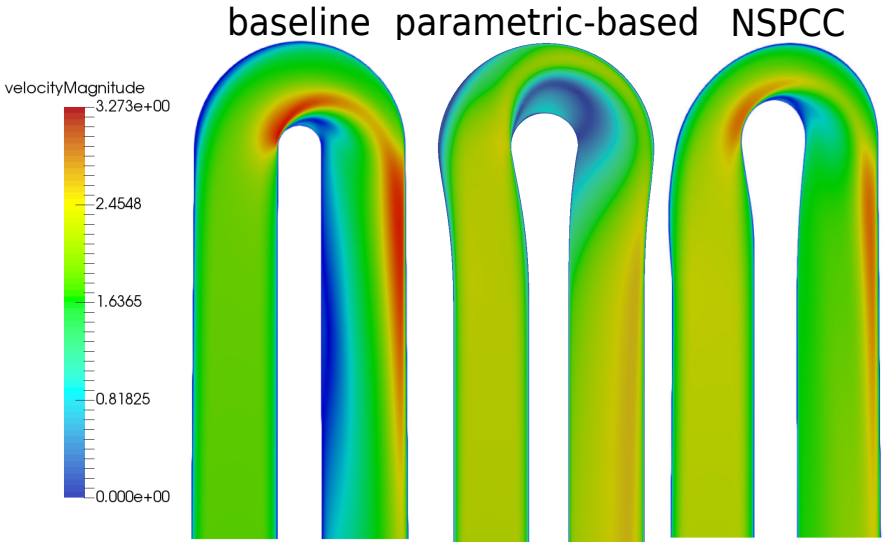


Figure 6.18: U-bend: comparison between the velocity magnitude of the optimal shapes provided by the parametric-based and the NSPCC parametrisations at mid height plane of the test case.

U_1, Sec. 6.5) and the diameter of the channel section (design mode number U_2, Sec. 6.5). The effects of these design modes allow a faster flow reattachment after the U-turn w.r.t. the baseline geometry. The main difference between the two optimised shapes relies on the fact that the flow separation generated by the optimal shape obtained with the parametric-based design is larger then the flow separation provided by the NSPCC results. This difference, which is the physical explanation of the 5.1% gap between the cost function's drop provided by the parametric-based design (18%) and the drop obtained with the NSPCC one (23.1%), arises from the fact that the parametric-based optimisation is not fully converged. Further research is needed to make the parametric-based shape optimisation completely converge (Sec. 6.5).

Fig. 6.19 provides the comparison of the static pressure distribution at mid height plane between the baseline and the optimised geometry (parametric-based and NSPCC parametrisations). This result highlights several similarities between the two parametrisations. Both of them reduce across the U-turn the pressure gradient between the inner and the outer wall. In both optimised shapes the low pressure area at the U-turn disappears. As result of this, the required pressure at the inlet is reduced to a similar magnitude value.

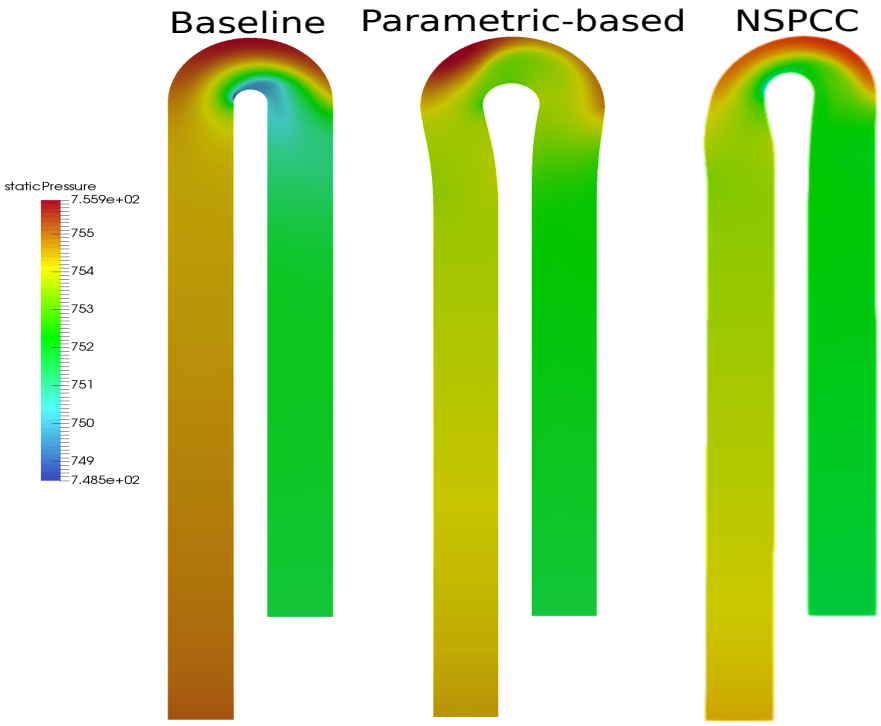


Figure 6.19: From left, comparison between the static pressure at mid height plane of the: (i) baseline geometry, (ii) optimal shape given by the parametric-based design and (iii) optimal shape provided by the NSPCC parametrisation.

6.7.2 Latest NSPCC results

A qualitative comparison can be performed between the optimal CAD models given by the three NSPCC parametrisations and the optimised parametric-based CAD model. It is possible to verify that a general agreement between these four CAD models can be identified. As for the parametric-based design, all the three optimal shapes given by the NSPCC parametrisations (i.e. L1, L2 and L3, Subsec. 6.6.2) reproduce the design modes U_1 and U_2 (Sec. 6.5). The surface undulations presented by the Opt-L3 shape are not presented by the parametric-based optimised shape. This difference is mainly due to the high refinement guaranteed by the level-3 parametrisation (1152 control points).

Fig. 6.20 provides a comparison between the velocity magnitude plot at mid height for all the optimised shape (parametric-based and NSPCC parametrisations). As for the NSPCC results shown in Sec. 6.7.1, the main difference between the parametric-based design and the NSPCC ones (Subsec. 6.6.2) relies on the flow separation which is generated by the optimal parametric-based design. A fully converged shape optimisation should be able to avoid such separation. Further research is needed to verify this (Sec. 6.5).

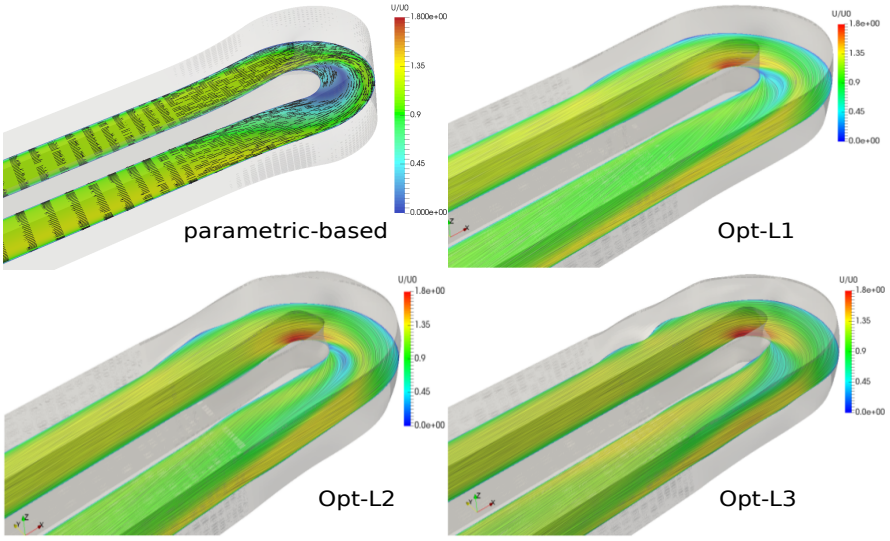


Figure 6.20: U-bend: comparison between the velocity magnitude at mid heigh of the U-bend optimised with the parametric-based design against the velocity magnitude at mid height given by the NSPCC parametrisations.

6.7.3 Discussion

As explained in Sec. 6.5, the parametric-based results provided for this test case are intermediate ones. However, these results can still be used to verify the reliability of the parametrisation approach proposed in Sec. 5.5. In terms of cost function reduction, the NSPCC approach is able to further reduce the cost function by 5/9% w.r.t. the parametric-based design. This difference is not negligible. On the other hand, the design space provided by the NSPCC approach is much more refined w.r.t. the parametric-based design. The parametric-based design is controlled by 48 control points (total number of design parameters: 96, Sec. 5.5) whereas the NSPCC approaches considers in total a set of control points which ranges between 288 to 1152. The reduced design space provided by the parametric-based design makes this design space much more easy to define and post-process w.r.t. the NSPCC approach.

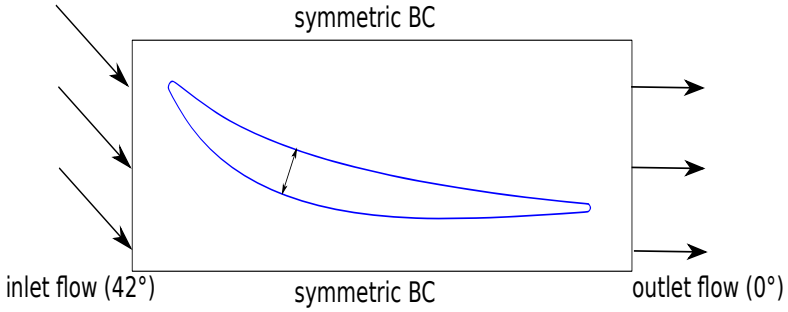


Figure 6.21: TUB: baseline flow conditions.

6.8 Flow optimisation of the TUB compressor stator blade

6.8.1 Simulation settings

The optimisation framework shown in Subsec. 6.2.2 is used to optimise also the TU Berlin CAD model (Sec. 5.8). The cost function to be minimised is the total pressure loss between the inlet and the outlet of the TU Berlin Stator, expressed as in Eq. 6.24:

$$\begin{cases} J = P_{total,inlet} - P_{total,outlet} \\ P_{total,S} = \frac{\int_S p_{total}\rho(u\cdot\hat{n})dS}{\dot{m}} \\ \dot{m} = \int_S \rho(u\cdot\hat{n}) \end{cases} \tag{6.24}$$

where $\dot{m}$ is the mass flow rate, ρ is the density, P_{total} is the total pressure and u represents the velocity. As for the U-bend, the CFD computations are performed by the in-house discrete adjoint solver STAMPS [145]. The parametrisation proposed in this thesis is explained in Sec. 5.8. It consists of a 2D section controlled by a set of user-friendly parameters such as the comber-line, a symmetric thickness distribution normal to the camber-line and the LE/TE radius. Every parameter is assigned with a law of evolution, which allows to determine the value of the parameter spanwise. It is therefore possible to construct a set of 2D sections which is then used to compute the final B-spline surface of the blade. The optimiser chosen is the limited memory BFGS algorithm with bound constraints (L-BFGS-B) already introduced in Sec. 4.3. The version of the L-BFGS utilised for this optimisation allows the imposition of all the manufacturing constraints (i.e. the minimum thickness) apart from the assembly ones.

6.8.2 Optimisation Constraints

The L-BFGS-B algorithm allows the definition of boundaries within which find the best set of design parameters. The constraints which have to be respected during the flow optimisation are stated in Sec. 5.7. These constraints expects: (i) to guarantee the G2 continuity along the overall 2D blade section, (ii) to keep constant the axial chord, (iii) to have a maximum value of the thickness distribution which is at least 8 mm, with the exception of the hub and shroud areas, where the accommodation of the assembly constraints (four mounting bolts) requires a maximum thickness distribution value of 10 mm and (iv) to impose that the minimum value of the LE and TE radii is 1 mm.
These constraints are respected during the flow optimisation as follows:

- *G2 continuity*: imposed along all the section based on the geometrical construction (which is met by construction by parametrisation, Sec. 4.2).
- Axial chord: the axial-coordinate of the last camber-line control point is set equal to the axial-coordinate of the first control point plus the constant axial chord value.
- Thickness distribution: the thickness between the suction and pressure surface is approximated using the corresponding B-Spline control point distances.
- LE and TE radii: the lower bound values are specified (the LE and TE radii are two design parameters of the optimisation, Sec. 4.2).

A first formulation of this test case prescribed also the imposition of the null exit angle as a constraint. This constraint has been considered unfeasible because a fixed exit angle would have has a too strong influence on the reduction of the cost function (the pressure losses can be effectively reduced by increasing the exit angle w.r.t. the inlet).
Several are the solutions which can be considered. One is to consider a multi-objective optimisation problem, where the cost function consists of two terms: (i) the ΔP and (ii) the squared value of the exit angle. If this cost function is considered, the goal of the optimisation is to identify the best compromise between the minimum pressure losses and the minimum mismatch of the exit angle [75].
In this thesis, the single-objective function (i.e. the ΔP) is considered. The blade is fixed at the hub and the shroud such that the optimiser is prevented from increasing the exit angle (i.e. unloading the blade) without any limitation.

6.9 TUB: initial flow field

The initial flow conditions are presented in Fig. 6.21. The compressor blade turns the flow from the inlet direction of 42° to the axial direction. The number of blades is fixed to 15. This allows to simplify the calculation from the overall stator to a single blade of the stator (i.e. to apply the periodic boundary condition). The gas is considered ideal with ratio of specific heat of 1.4. The boundary conditions are set as follows:

- inlet: total pressure (102713.0 Pa), total temperature (294.314 K), whirl angle (42°), pitch angle (0°) and turbulence intensity (4%).
- Outlet: the back pressure is identified in order to keep constant the mass flow rate (9.5 kG/s).

All the boundary conditions are constant values over the radial height.

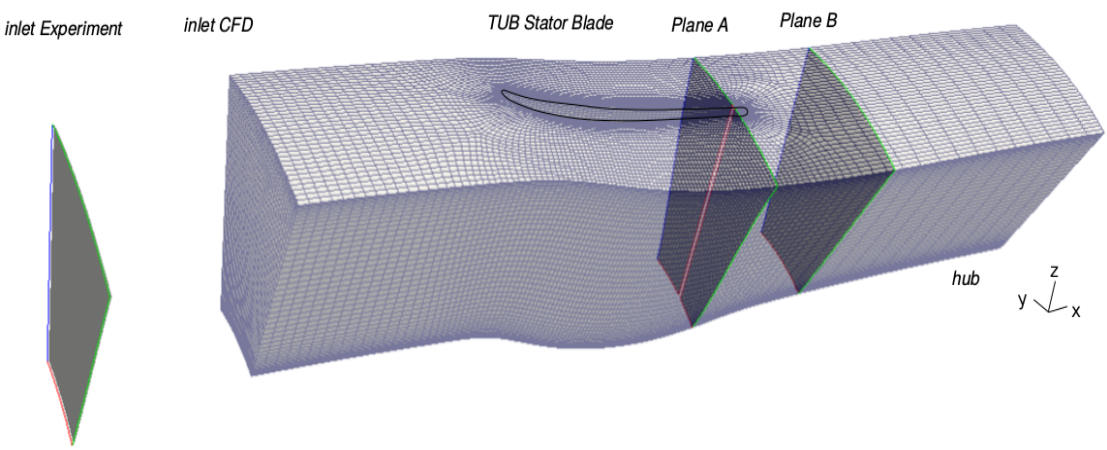


Figure 6.22: TUB: mesh with 400k cells and experimental settings [151].

Other researchers [151, 152] at CFD optimisation group of QMUL have analysed three meshes, which consist of: (i) 400k cells, (ii) 800k cells, (iii) 1.9M cells. The results provided by these meshes have been compared to the experimental investigations conducted by researchers at TU Berlin and Rolls Royce Germany [153]. Despite the different location of the experimental inlet w.r.t. the mesh one (Fig. 6.22), the simulation results can still be compared. Mykhaskiv [152] shows that both the fine mesh (1.9M) and the mid size mesh (800k) are able to reproduce flow conditions very close to the ones evaluated by the experimental data. The coarse one (400k cells) does not predict correctly such flow fields.

In this chapter the author will present simulation results obtained by utilising the coarse mesh (400k cells, Fig. 6.22). This simulation respects all the manufacturing constraints with the only exception of the assembly ones. This optimisation is an intermediate step which aims to demonstrate the applicability of the fully differentiated design chain to this test case. Thereafter, once verified, the optimisation which respects also the assembly constraints will be investigated in chapter 7 with the 800k cells mesh.

6.10 TUB: low fidelity results

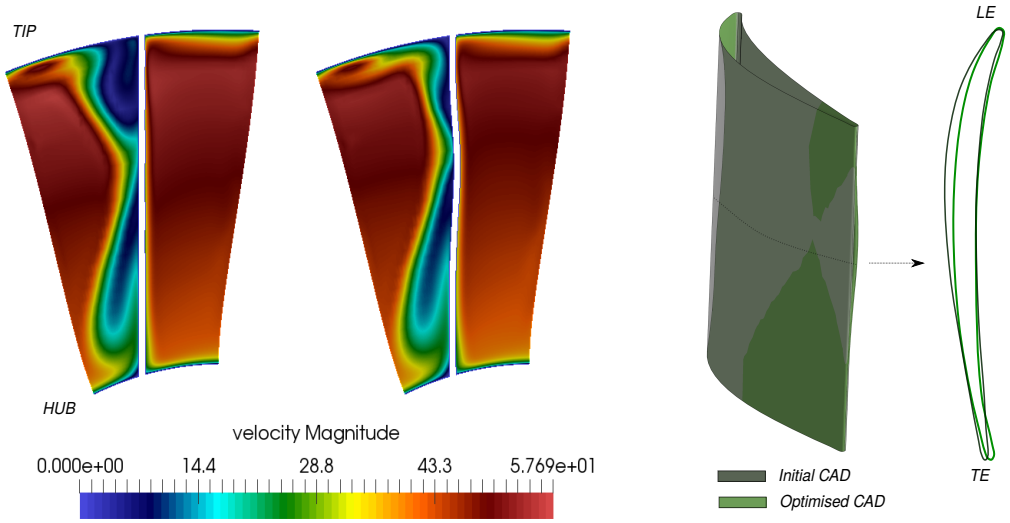


Figure 6.23: Baseline and optimised blade results at the trailing edge, on the left. Difference in CAD geometries and mid-span profile comparison, on the right.

After twenty-five iterations, the optimiser had modified the CAD parameters which resulted in an overall thinner blade and a leaning trailing edge at mid-span. The optimised blade improves objective function by 7% and reduces the tip flow separation significantly, as shown in the Fig. 6.23 and Fig. 6.24. These flow results can not be considered completely reliable. The computational grid used to perform this low-fidelity optimisation does not correctly predict the initial flow field. Moreover, these results do not respect the assembly constraints.
The main contribution given by these results to this thesis is that they verify the applicability of the fully differentiated design chain (and of the proposed blade parametrisation) to the TUB test case. The implementation of the assembly constraints will require a different optimiser and a specific approach developed by

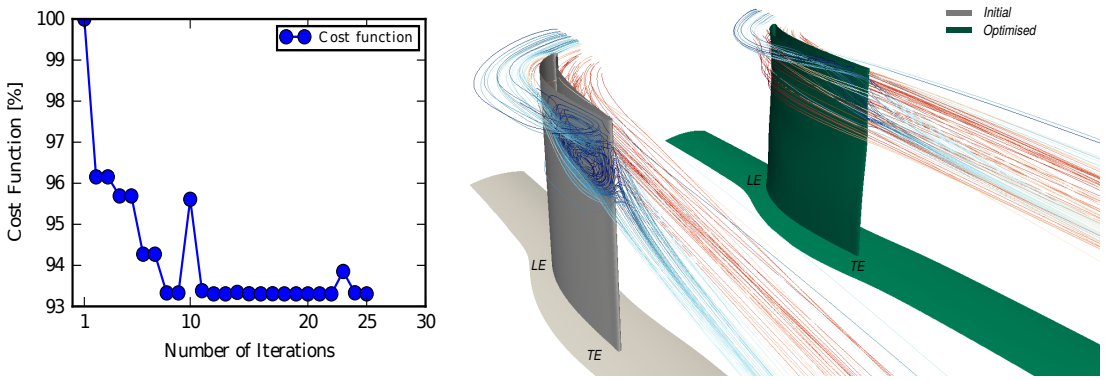


Figure 6.24: Cost function history, on the left. 3D representation of the tip vortex reduction from the baseline to the optimised blade, on the right.

using the CAD features provided by the CAD kernel. Also, the mesh needs to be upgraded to the mid-size one (800k cells) in order to correctly predict the initial flow field.
All this will be investigated in the following chapter.

6.11 Summary

This chapter has presented the *adjoint CFD method*, which allows to efficiently perform the calculation of the flow derivatives (Sec. 6.1). STAMPS is the CFD/adjoint CFD solver used in this research and developed by the CFD optimisation group of QMUL. The CFD derivatives computed by STAMPS have been plugged into OCCT differentiated in reverse mode such that a fully differentiated design chain has been finally defined (Sec. 6.2). This fully differentiated design chain efficiently and robustly computes the total derivatives of the cost function w.r.t. the design parameters of the CAD model.
The design chain has been used to perform the aerodynamic shape optimisation of the parametric CAD models explained in Ch. 5: a cooling channel for high pressure turbine blades (Sec. 5.5) and a compressor stator blade (Sec. 5.8). The optimisation results obtained for the U-bend are intermediate ones. The shape optimisation (Sec. 6.5) has not been completed because the deformed mesh has become too poor for the solver to converge. However, it is still possible to compare the results given by the parametrisation proposed in this thesis against the ones provided by the reference parametrisation, which is the NSPCC approach (Sec. 6.6). This comparison (Sec. 6.7) allows to verify that the parametrisation approach proposed in this thesis is able to reproduce the main design modes considered by the optimal shape given by the NSPCC parametrisation. The reference

parametrisation performs better in terms of cost function reduction. This is mainly due to the fact that the NSPCC approach provides a much higher refinement of the design space. On the other hand, the parametric-based design proposed in this thesis is much more suitable to the typical workflow of the designer, who prefers to handle intuitive and user-friendly parameters to define the geometry and pre-/post-process the results.
The compressor stator blade has been optimised by using a coarse mesh (400k cells, Sec. 6.10). Despite this optimisation not respecting the assembly constraints, the results shown in Sec. 6.10 allow to verify that the fully differentiated design chain can be used to successfully complete the shape optimisation of the TUB test case. The optimisation of this test case while embedding all the constraints and utilising a finer mesh (800k cells) which is able to correctly reproduce the initial flow field will be presented in Ch. 7.

Chapter 7

Optimisation of TUB test case with embedded assembly constraints

7.1 Introduction

An optimised component is normally assembled with other components of the final product by threaded fasteners such as screws, nuts and bolts. These assembly constraints should be added to the shape optimisation problem. The optimised component which does not respect the assembly constraints needs to be modified after having performed the optimisation. This modification would result in several disadvantages. First, additional efforts (e.g. additional CFD analyses) would be necessary to verify the performance of the final geometry. Second, the modification could affect the performance of the final geometry. This would imply another loop of optimisation and, therefore, other time spent to identify the optimal geometry. In this chapter, the geometry of the TUB test case is aerodynamically optimised. This test-case has been investigated in several earlier works [75, 85, 154]. None of them embeds the assembly constraints (i.e. considers the insertion of four mounting bolts, two at the hub and two at the shroud) while performing the gradient-based shape optimisation. The approach utilised by Mykhaskiv et al. [85] could allow to fulfill such requirement. They impose the geometrical constraints with implicit parametrisation based on the BREP description (NURBS) and respect the thickness and the leading/trailing edge radius constraints of a compressor stator blade by using discrete spaces constructed with test-points. This approach has still to be tested to fit the assembly constraints.

The most promising approach is to impose that the area of intersection between the blade and the assembly constraint is never positive. This would allow to impose such constraint also for complex geometries with several points of contac-

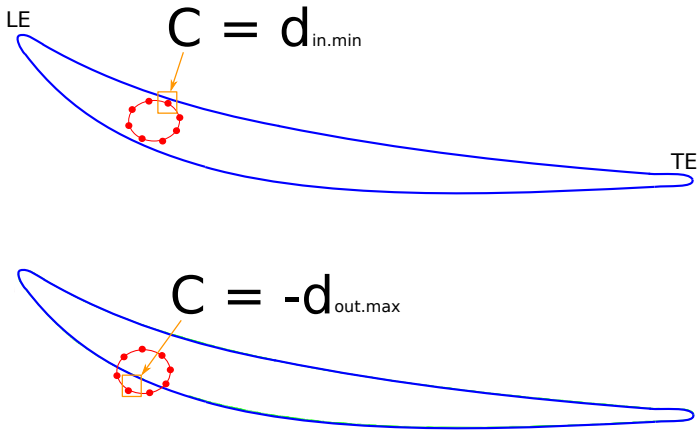


Figure 7.1: Example of constraint satisfied (top)/violated (bottom) by using the signed distance approach used in this thesis.

t/minimal distances, such as a duct in a complex engine bay. The application of this approach requires the differentiation of intersection algorithms implemented within a CAD system. As shown in the following sections, this differentiation is not straightforward, i.e. a smooth intersection area function is not easy to identify. This makes this approach unreliable for such purpose.
Armstrong et al. [19] aerodynamically optimise the shape of an automotive duct while respecting the constraints on packaging space due to surrounding components. After having investigated the algorithms offered by the commercial CAD software CATIA, they have verified that the only algorithm which could be used to respect such constraints is the interference one. They compute the distance between the surrounding components and the duct and impose that this distance never becomes negative, i.e. no interference is detected.The derivatives of the interference function w.r.t. the CAD parameters are computed using the FD. This approach has still to be verified with a test case such as the TUB, which requires the movement of the assembly constraints during the flow optimisation.

The approach utilised in this thesis (Fig. 7.1) evaluates a signed distance to the component to be optimised (i.e. the blade) of sampled points on the assembly constraint (i.e. the cylinder where the bolt has to be accommodated). The assembly constraint function (C in Fig. 7.1) is implemented by using the differentiated OCCT. Therefore, the derivatives of the assembly constraint function are computed with the AD technique.
In other words and summarising, one of the main bottlenecks that limits the spread of gradient-based shape optimisation chains in engineering design is the difficulty to provide optimised shapes which respect all the manufacturing constraints. In

particular, the identification of a smooth assembly constraint function is difficult. The interference approach of Armstrong et al. demonstrates that an aerodynamic shape optimisation chain which uses FD to differentiate assembly constraints can be utilised to get such goal. On the other hand, the signed distance approach implemented in this thesis reaches the result of differentiating the assembly constraints function while allowing the movement of these constraints during the flow optimisation by using a fully differentiated design chain, i.e. by computing the assembly constraint derivatives with the automatically differentiated OCCT.
In Sec. 7.2, the author gives the geometrical details about the insertion of the four mounting bolts and, also, explains the approaches which can be used to fit the assembly constraints during the flow optimisation. Among these approaches, three of them utilise the intersection algorithm implemented within OCCT. The forth one is the signed distance approach shown in Fig. 7.1. Sec. 7.3 gives the first geometrical application that is used to test the three approaches which utilise the OCCT intersection algorithm. Sec. 7.4 provides a second geometric application that serves to verify the reliability of the intersection approaches in gradient-based optimisation problems. Sec. 7.5 presents the implementation of the signed distance approach, which is finally used to impose the assembly constraints during the flow opimisation. Sec. 7.6 provides the results of the CAD-based shape optimisation of the TUB with all the constraints embedded in the flow optimisation. Sec. 7.7 gives the optimisation results provided by a refined parametrisation. These results are compared to the ones shown in Sec. 7.6.

7.2 Imposition of the assembly constraints

The TUB test case expects the accommodation of four mounting bolts (which occupy the space of four cylinders), two at the hub and two at the casing (Fig. 7.2). The two cylinders for the fixture in the middle of the blade have a radius of 2.5 mm and a depth of 20 mm. The blade thickness at these positions has to accommodate a cylinder (radius = 5 mm, depth = 20 mm) to allow cutting of the thread at both hub and casing, Fig. 7.2. The two cylinders can be placed arbitrarily inside the profile shape, but have to be at least 60 mm apart from each other. The optimisation framework is the same as explained in Subsec. 6.2.2 (shown in Fig. 6.2). The optimiser chosen is the Sequential Quadratic Programming (SQP) (introduced in Subsec. 7.2.1).

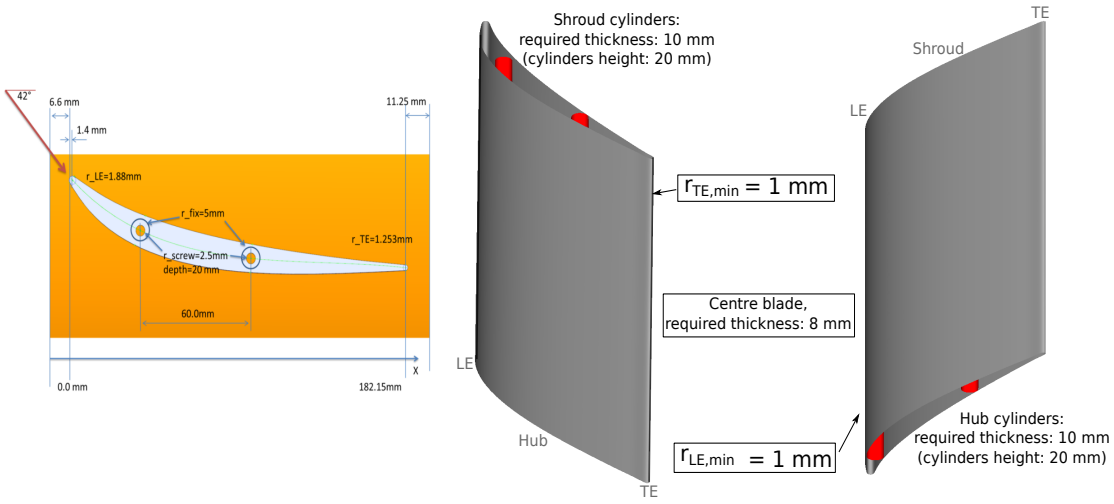


Figure 7.2: TUB constraints during flow optimisation.

7.2.1 The SQP Method

The Sequential Quadratic Programming (SQP) is one of the most successful methods for the numerical solution of constrained nonlinear optimization problems. Defined a nonlinear optimisation problem (NLP) in the form:

$$min\, f(x) \qquad \text{with } x \in \mathbb{R}^n \tag{7.1}$$

subject to:

$$\begin{cases} H(x) = 0 \\ C(x) \geq 0 \end{cases} \tag{7.2}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $H : \mathbb{R}^n \to \mathbb{R}^m$ and $C : \mathbb{R}^n \to \mathbb{R}^p$ are the equality and the inequality constraints.
The SQP is an iterative procedure which models the NLP (expressed by Eq. 7.1 7.2) for a given iterate x^k, $k \in \mathbb{N}_0$ by a Quadratic Programming (QP) subproblem, solves this QP subproblem, and then uses the solution to construct a new iterate x^{k+1}. The sequence $(x^k)_{k\in\mathbb{N}_0}$ converges to a local minimum x^* of the NLP for $k \to \infty$. A detailed explanation of the SQP method that is used in this thesis can be found here [155]. The SLSQP implementation used here is Python version 3.6 included in SciPy[1].

[1] `https://docs.scipy.org/doc/scipy-0.17.0/scipy-ref-0.17.0.pdf`

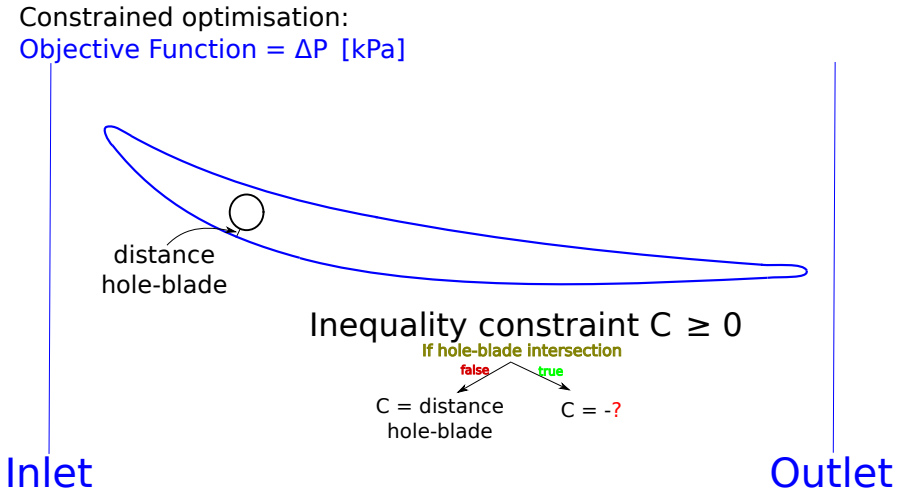


Figure 7.3: Example of TUB flow optimisation with assembly constraints in the optimisation loop.

7.2.2 Possible approaches to impose the assembly constraints

The utilisation of the SLSQP optimiser allows the imposition of bounds to the design parameters. These bounds are a separate set of constraints specifically for the design variables which are used in this thesis to respect the manufacturing constraints mentioned in Subsec. 6.8.2. Moreover, as mentioned in Subsec. 7.2.1, this optimiser enables the imposition of inequality constraints ($C(x) \geq 0$ in Eq. 7.2). This property can be used to respect the assembly constraints. To simplify the explanation, one cylinder (i.e. one bolt) will be considered in this subsection.

One can define the value of C based on the position of the hole w.r.t. the blade. Given the settings in Fig. 7.3 (i.e. the hole inside the volume of the blade), the value of the inequality constraint C is equal to the minimal distance between the blade and the cylinder. Considering that the SLSQP optimiser evaluates the constraint as satisfied if its value is positive, the condition $C \geq 0.$ is respected. When the intersection between the cylinder and the blade occurs, the minimal distance between two shapes is null. For this reason, the constraint C has to be defined with another value. If the intersection occurs, the sign of the constraint has to be set as negative, such that the constraint is violated (the constraint imposed by the SLSQP is: $C \geq 0$). The magnitude of the constraint C can be computed by using several approaches. The approaches considered in this thesis are as follows:

1. the topological intersection approach (AC1).
2. The surface-surface intersection approach (AC2).
3. The 2D approach with curve-curve intersection (AC3).

4. The signed distance approach (AC4).

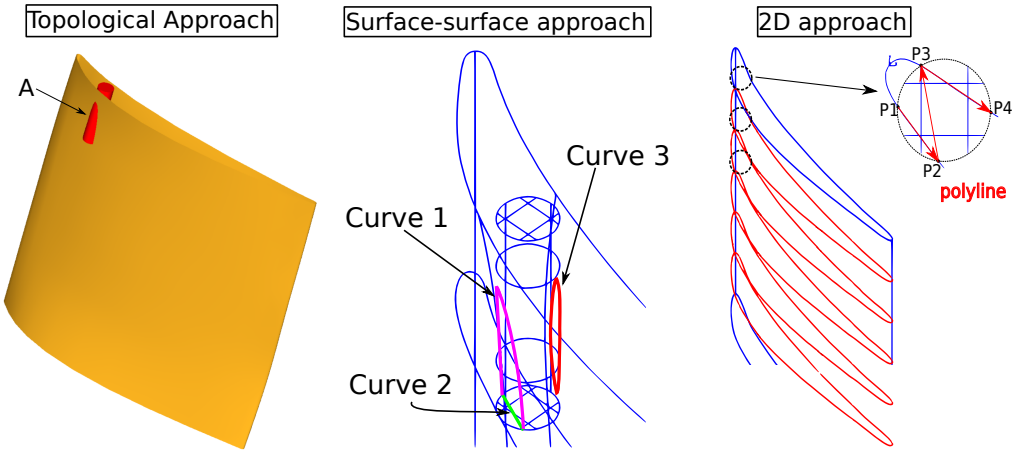


Figure 7.4: Possible approaches to compute the constraint C when the intersection blade-cylinder occurs.

The topological intersection approach computes the area of intersection ("A" in Fig. 7.4) between the cylinder and the blade at topological level by using `BRepAlgoAPI_Section`[2], which is the topological intersection algorithm provided by OCCT. This algorithm takes as input two objects (i.e. the blade and the cylinder) and analyses their topological properties (relative position of faces and edges that compose each object). Then, the intersection between the neighbour faces of the two objects is calculated and its area is computed. The implementation (simplified) of the function used to compute the area of intersection with this approach is shown in Appendix C, code listing C.1.
The surface-surface intersection approach (Fig. 7.4) considers the CAD models of the blade and of the cylinder and extracts from these CAD models the corresponding surfaces. Then, the curves of intersection between the surfaces of the cylinder and the surfaces of the blade are computed by using the algorithms given by OCCT to compute the area of intersection between surfaces (`GeomAPI_IntSS`). The intersecting curves are finally used to compute the area of intersection on the surface of the blade. The implementation of this approach is presented in Appendix C, code listing C.2.
The 2D approach with curve-curve intersection (Fig. 7.4) utilises the algorithm of OCCT which computes intersections between 2D curves. Since a cross-sectional design approach is used to build the blade, for every 2D section of the blade, the

[2] `https://www.opencascade.com/doc/occt-6.9.0/refman/html/class_b_rep_algo___section.html#details`

2D circle (which represents the 2D section of the cylinder) is constructed. Then, for every 2D section, the intersection between the circle and suction/pressure 2D B-splines can be computed, as shown in Fig. 7.4. Finally, the distance between all adjacent points is calculated and added to the total distance value. A simplified implementation of the function used to compute this distance (`polylineLength`) per every slice of the blade that is intersecting the cylinder is reported in Appendix C, code listing C.3.

Defined as $lowerBoudaryCylSlices$ ($LBCS$) and $upperBoudaryCylSlices$ ($UBCS$) the lower and the upper slices of the blade that are intersecting the hole, the final value of the constraint C can be computed as in Eq. 7.3:

$$C = \sum_{i=LBCS}^{UBCS} polylineLength_i \tag{7.3}$$

The last approach considered by the author is the signed distance approach, already introduced in Sec. 7.1. This approach defines the cylinder as a point cloud and identifies the set of points of the cloud which are outside of the volume of the blade. Among these points, the one at maximum distance from the blade is identified and this distance is equalised to the magnitude of the constraint value C (Fig. 7.1).

The approaches which are considered by the author the most promising are the intersection ones (AC1, AC2, AC3). These approaches should allow to respect the assembly constraints also for complex geometries with several points of contact /minimal distance. Sec. 7.3 will apply these approaches to geometric optimisation problem, i.e. optimisation problems that do not consider the flow during the optimisation. These optimisation problems, whose resolution requires the calculation of the derivatives of the assembly constraint w.r.t. the design parameters of the optimisation, will allow to test these approaches (AC1, AC2 and AC3) in gradient-based problems.

The signed distance approach (AC4) is considered a backup option. It is the most immediate approach to fit the assembly constraints. However, its extension to more complex geometries does not seem easy to achieve. For this reason, it will be used if AC1, AC2 and AC3 will be verified not to be reliable solutions.

7.3 Geometric application I: hole fitting

This section provides the application of the intersection approaches proposed in Subsec. 7.2.2 (i.e. AC1, AC2, AC3) to the first geometrical optimisation problem

studied in this thesis. The approaches which will provide the most promising results will be applied to the geometric optimisation explained in Sec. 7.4.

7.3.1 Optimisation settings

The first geometric application considers a blade optimised without respecting the assembly constraints such that the cylinder does never fit completely in the blade without intersecting. The goal of the optimisation is to find the position of the cylinder such that the cost function (intersection area A, Eq. 7.4):

$$min\ A(x, y) \tag{7.4}$$

is minimal by varying the (x, y) coordinates of the centre of the cylinder, as shown in Fig. 7.4. In order to solve this optimisation problem, the derivatives 7.5

$$\nabla A = \left(\frac{\partial A}{\partial x}, \frac{\partial A}{\partial y} \right) \tag{7.5}$$

are computed by using the differentiated **OCCT**. This optimisation does not require the definition of inequality constraints. Therefore, the limited-memory BFGS introduced in Sec. 4.3 is employed as optimiser.

7.3.2 AC1: topological intersection approach

The optimisation has not converged. The reason for this can be evaluated in the plot on the right in the Fig. 7.5. In this figure, the plot of the objective function (area of intersection between the cylinder and the blade) by varying the (x, y) of the centre of the cylinder is presented. This plot is obtained by considering a set of (x_i, y_i) (the coordinates of the centre of the hole), with $i = 1 \dots 10000$, and by computing per every (x_i, y_i) the relative value of the area of intersection. This area plot presents several erroneous drops. Each of these drops corresponds to a value of the area which is incorrectly calculated.
The erroneous drops in the area plot (highlighted with the red circles) have affected the optimisation convergence. The main reason that justifies the presence of these drops is that `BRepAlgoAPI_Section` works poorly in tangential cases [156]. An example of tangential case is the "B" side shown in Fig. 7.5. The commercial CAD tools are normally equipped with a second intersection algorithm specifically dedicated to compute such tangential cases. This second algorithm is not provided by **OCCT**.

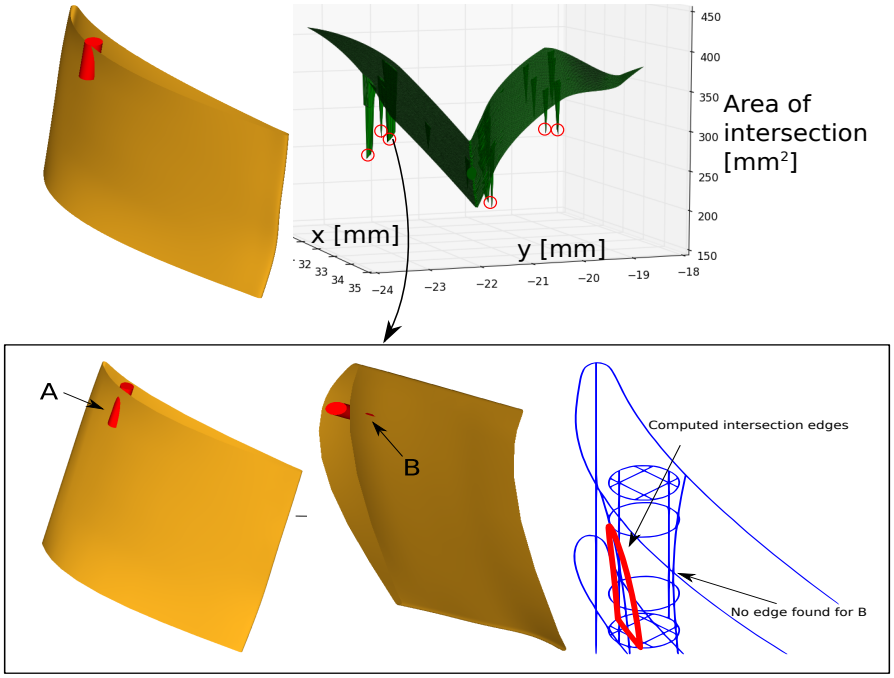


Figure 7.5: Optimised blade and a cylinder (left) and area of intersection between the cylinder and the optimised blade by varying the (x; y) coordinates of the centre of the cylinder (right). One of the erroneous area of intersection value is highlighted: `BRepAlgoAPI_Section` failed to compute "B" side.

Finally, the intersection values and the derivatives were incorrectly computed, as shown in Fig. 7.5, such that the optimisation could not converge.

7.3.3 AC2: surface-surface intersection approach

Fig. 7.6 shows the plot of the area of intersection between the blade and the cylinder computed by using the AC2 approach (Subsec. 7.2.2). This area plot is calculated by using the same set of (x_i, y_i) coordinates of the centre of the cylinder. The derivatives 7.5 (∇A) are computed by using the differentiated **OCCT** and the optimisation successfully converged (Fig. 7.7). It is necessary to highlight that the convergence of this fitting problem does not imply that the surface-surface intersection approach is a reliable solution for a gradient-based optimisation. As Fig. 7.6 demonstrates, the cost function still presents several erroneous drops (i.e. for some (x, y) coordinates of the centre of the cylinder, the area of intersection between the cylinder and the blade is not correctly computed) which make this approach unreliable for the purpose of this research.

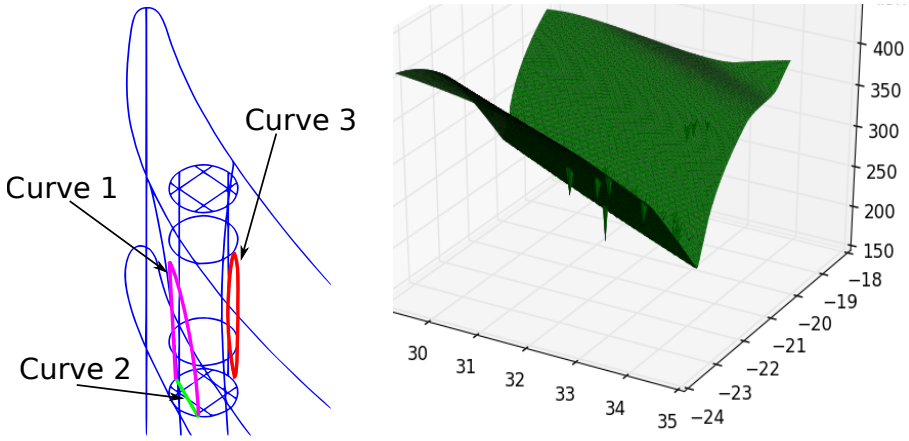


Figure 7.6: surface-surface intersection approach for the evaluation of the area of intersection between the cylinder and the blade.

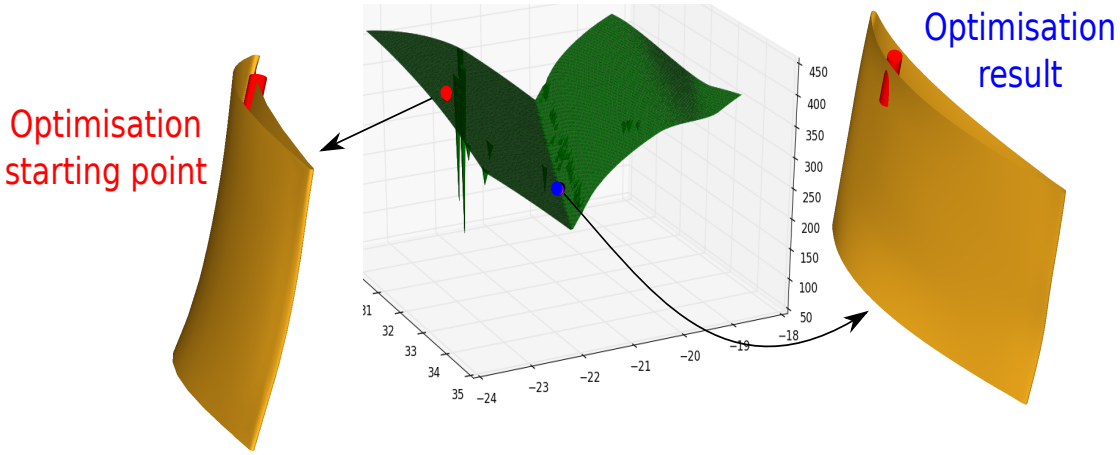


Figure 7.7: Fitting problem by using the geometrical class provided by **OCCT** to compute the area of intersection between the blade and the cylinder.

7.3.4 AC3: 2D approach with curve-curve intersection

The AC3 approach (Subsec. 7.2.2) computes the constraint value as in Eq. 7.3. By using this approach, the optimisation successfully converges, as shown in Fig. 7.8.

Moreover, the cost function plot does not present any erroneous drops, which

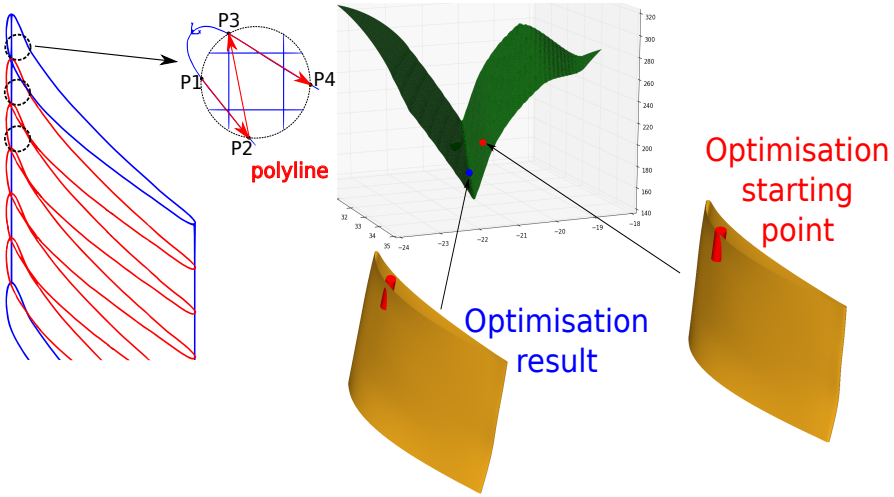


Figure 7.8: 2D slices approach (left) and optimal position of the cylinder evaluated with 2D approach (right).

could affect the calculation of the derivatives. Therefore, the AC3 approach can be a reliable solution for the implementation of the assembly constraints during the flow optimisation and will be investigated further with the second geometric application explained in Sec. 7.4.

7.3.5 Discussion

The application of the three approaches AC1, AC2 and AC3 to the geometric application studied in this section has allowed to verify that the topological intersection algorithm given by OCCT (AC1 approach) can not be applied to fit the holes in a gradient-based shape optimisation. This is due to the primal given by this approach (Sec. 7.3.2), which presents erroneous drops that affect the computation of the derivatives.

The AC2 approach allows to solve the optimisation problem considered in the geometric application I. However, its extension to more complex optimisation problems (e.g. which consider an higher number of design parameters) seems difficult. This is due to the fact that also this approach computes erroneously some values of the primal (Sec. 7.3.3).

The AC3 approach is the only one that is specifically implemented for cross-sectional parametrisations. This makes its extension to other types of parametrisation difficult to achieve. However, this is also the approach which has provided the most encouraging results. It allows to solve the optimisation problem given in the geometric application I and correctly computes the primal (Subsec. 7.3.4). For this reason, it will be investigated with a second geometric application (Sec. 7.4).

7.4 Geometric application II: blades fitting

A geometrical application that serves as preparation step for the execution of a workflow that includes also a flow solver is defined. This geometrical application is a typical, often executed task in CAD, so-called "surface fitting" (Sec. 4.3). This application (Subsec. 7.4.1) is a step towards the flow optimisation because it considers the cylinder as a constraint which has to be respected by using or the AC3 (Subsec. 7.4.2) or the AC2 approach (Subsec. 7.4.3). The outcome of these optimisations is discussed in Subsec. 7.4.4.

7.4.1 Optimisation settings

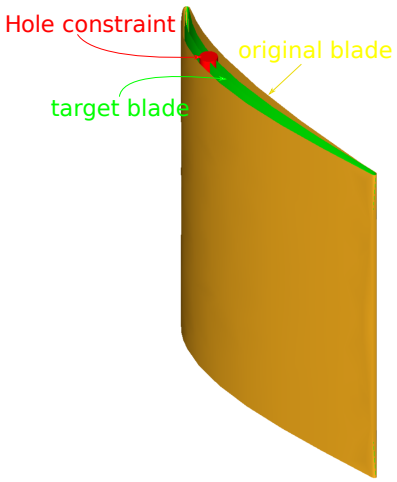


Figure 7.9: Original (yellow) and target (green) blade with the cylinder constraint (red).

This optimisation considers two blades: P, which is controlled by the set of design parameters X_j ($j = 1 \dots 184$, Sec. 5.8) and a target geometry T (Fig. 7.9). The goal of this optimisation is to minimise the total distance $f(x)$ between P

and T, Eq. 7.6:

$$\min_{x \in \mathbb{R}^{184}} f(x) = \sum_{i=1}^{20000} |P_i(x_j) - T_i|^2 \quad j = 1, \ldots, 184. \tag{7.6}$$

where j is the number of design variables, i is the index of one of 20000 sampling points distributed uniformly over the two blades, $P_i(x)$ and T_i are the points on the original (yellow surface in Fig. 7.9) and target (perturbed surface in Fig. 7.9) surfaces respectively.
The optimisation case proceeds as follows:

1. construct two blades: original and target (green) blade together with a cylinder, as shown in Fig. 7.9.

2. Sample both final B-spline surfaces with 20K pairs of (u_i, v_i) parametric coordinates. These parametric coordinates are later used in B-spline algorithms to evaluate the corresponding three-dimensional points $P_i(x)$ and T_i.

3. Define the objective function as in Eq. 7.6.

4. Define a constraint (by using the AC3/AC2 approach) to consider the intersection value between the original (yellow) blade and the cylinder.

5. Declare the original design parameters (x) which control the original blade as independent variables of the optimisation problem.

6. Minimise the objective function by using the SLSQP optimisation algorithm from *SciPy* library provided by Python.

The SLSQP feature of considering additional equality/inequality constraints during the optimisation enables the integration of the cylinder (i.e. the hole where the bolt has to be inserted) as inequality constraint.
After both parts are computed, the value $C(x)$ is provided to the optimiser as in Eq. 7.7:

$$C(x) = \begin{cases} -\sigma(x), & \text{if intersection exists.} \\ d(x), & \text{otherwise.} \end{cases} \tag{7.7}$$

where $d(x)$ is the distance between the original blade and the cylinder and $\sigma(x)$ is the intersection value between the original blade and the cylinder.
The SLSQP optimiser considers a constraint satisfied if the constraint value is non-negative. This is the reason for multiplying $\sigma(x)$ with -1 in Eq. 7.7 to indicate the constraint violation. Two are the possible solutions considered to compute $\sigma(x)$: the AC3 approach and the AC2 one (Subsec. 7.2.2).

7.4.2 AC3: 2D approach with curve-curve intersection

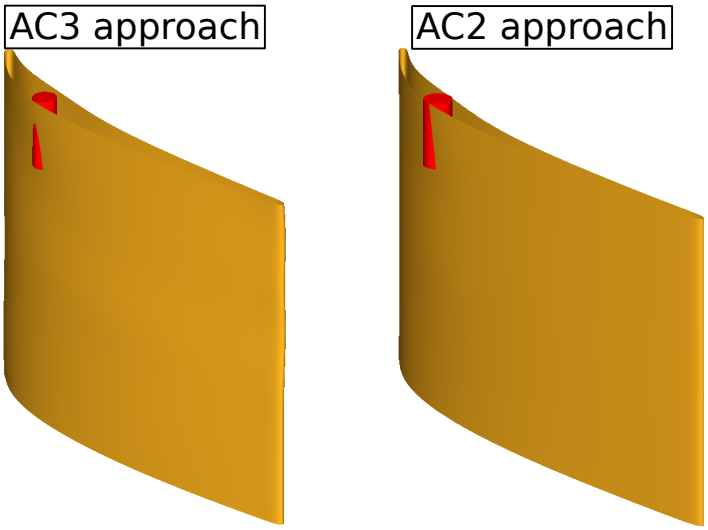


Figure 7.10: AC3 approach at iteration 90 (left) and the AC2 approach (right) at iteration 50. Both optimisations do not converge.

The 2D approach with curve-curve intersection method has already been successfully applied in Subsec. 7.3.4. In this section, the 184 design parameters of the blade are considered as independent variables, instead of (x, y) coordinates of the centre of the cylinder (Subsec. 7.3.1). This implies that the derivatives 7.8:

$$\frac{\partial C}{\partial x_i} \qquad \text{with } i = 1 \dots 184 \tag{7.8}$$

have to be computed, where C (defined in Eq. 7.3) is set as the inequality constraint value σ in Eq. 7.7.
During the optimisation process (cost function in Eq. 7.6), the intersection constraint was violated and the SLSQP optimiser was not able to completely recover it. The result is shown in Fig. 7.10. The reason of this failure is not easy to identify. The SLSQP optimiser used in this research is provided by the SciPy library, which is used as black-box. The analysis of the implementation of this optimiser would require extensive efforts and an high amount of time. This research does not provide the time necessary to perform such analysis or to implement another version of this optimiser. One of the next steps of this research is to identify the reasons of this failure.

7.4.3 AC2: surface-surface intersection approach

As already mentioned in Subsec. 7.3.3, the non-differentiable function of the area of intersection provided by the AC2 approach inhibits the utilisation of this approach to fit the assembly constraints during the flow optimisation.
Despite this, the author prefers to provide in this thesis a complete investigation of the problem. For this reason, the fitting problem of the geometrical application II is performed.
As expected, the fitting problem explained in Sec. 7.4 does not converge. The main reason of this failure is considered by the author the non-differentiable primal provided by the AC2 approach. In Fig. 7.10 (right) this optimisation result is shown.

7.4.4 Discussion

The investigation shown in Sec. 7.3 and Sec. 7.4 about the three approaches AC1, AC2 and AC3 has not given reliable results. For this reason, the author will utilise the signed distance approach AC4 to impose the assembly constraints during the flow optimisation.
However, the author considers this as an interesting investigation about automatically differentiated intersection algorithms. The main outcomes of this research are:

1. the unsuccessfully results given by the differentiated intersection algorithms are mainly related to the difficulty in defining a differentiable primal, i.e. a primal which does not present erroneous calculations that affect the computation of the derivatives.

2. The 2D intersection algorithm (AC3 approach) seems to provide more encouraging results. This should be related to its simplified implementation w.r.t. the 3D intersection classes implemented within OCCT (AC1 and AC2 approaches). Further research is needed to investigate the reasons of its failure in the geometric application II.

7.5 Signed distance approach

As verified in sections 7.3 7.4, all the intersection classes provided by **OCCT** do not allow to fit the assembly constraints during gradient-based optimisation. For this reason, the AC4 approach will be used to get this goal. Subsec. 7.5.1 gives the

details about the implementation of the signed distance approach. Before using this approach in the high fidelity optimisation of the TUB (Sec. 7.6), it has been successfully applied to a low fidelity simulation of the TUB with embedded flow.

7.5.1 Implementation of the signed distance approach

An approach based on the evaluation of a signed distance to the component to be optimised (i.e. the blade) of sampled points on the assembly constraint (i.e. the cylinder) is implemented. This approach is divided into two steps. Firstly, the

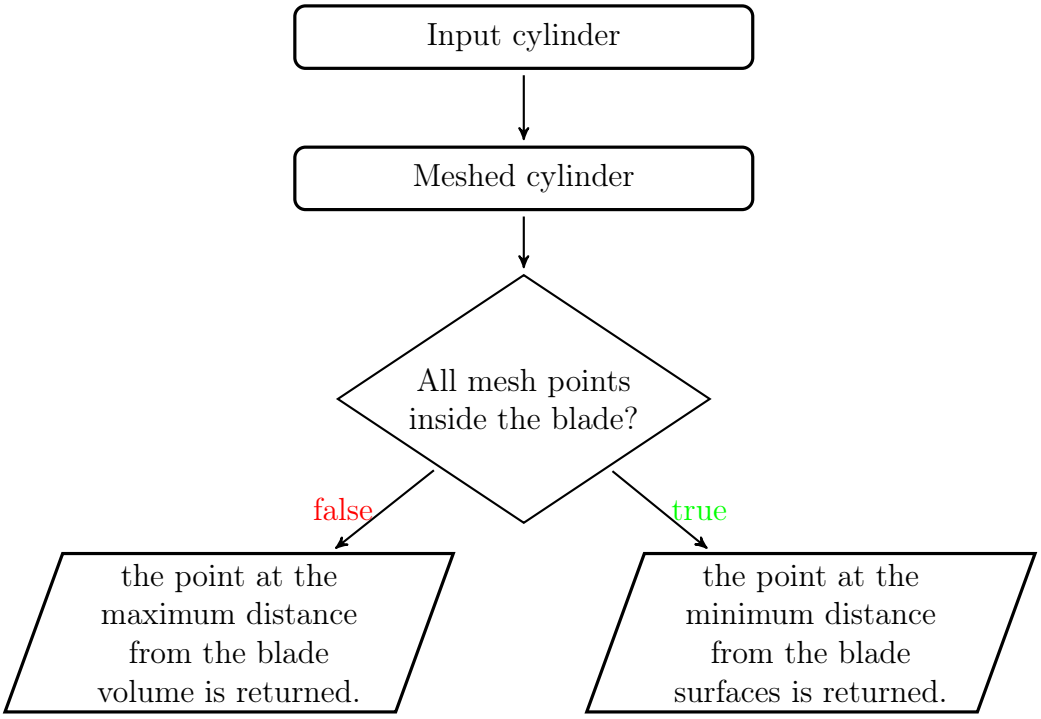


Figure 7.11: Implementation of `cylAnalyser` function.

function `cylAnalyser` (diagram shown in Fig. 7.11) is implemented. This function takes as input the cylinder and returns as output a point. In order to identify the point to return ("P"), the function follows the following steps:

- mesh the input cylinder.
- Evaluate whether all the mesh points are inside the volume occupied by the blade.
- If all the mesh points are inside the volume of the blade, `cylAnalyser` returns the point at the minimum distance from the blade.

- If some of the mesh points are outside of the volume occupied by the blade, `cylAnalyser` returns, among the points that are outside of the volume of the blade, the one at the maximum distance from the blade.

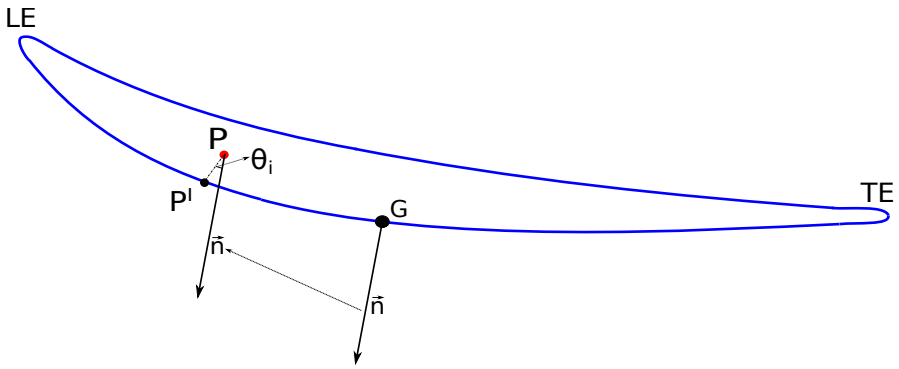


Figure 7.12: A possible approach to verify if a point is inside the 2D profile of the blade (suction side). The same approach is implemented also for the pressure side.

To verify if a mesh point P is inside the volume of the blade, the author has implemented a function called `isInside`. This function defines the following steps:

1. Identify the barycentric point G of the suction/pressure surface of the blade.
2. Construct the normal to the suction/pressure surface ($\vec{n}$ in Fig. 7.12) in the barycentric point.
3. Project the point P onto the suction/pressure surface. The projection of P is P'.
4. Compute the $cos(\theta_i)$ as in Fig. 7.12.
5. If $cos\theta_i \geq 0.$ for both suction and pressure surfaces, it is verified that the point P *is inside* the volume of the blade.

Then, a second function, called `constraintValue`, is implemented (diagram in Fig. 7.13). This function takes as input the point "P" returned by `cylAnalyser` and provides as output the constraint value. It is useful to remind that the SLSQP optimiser considers a constraint satisfied if its value is non-negative, as already mentioned in the Subsec. 7.2.1. For this reason, the returned constraint value is computed as follows:

- verify if the input point "P" is inside the volume of the blade.

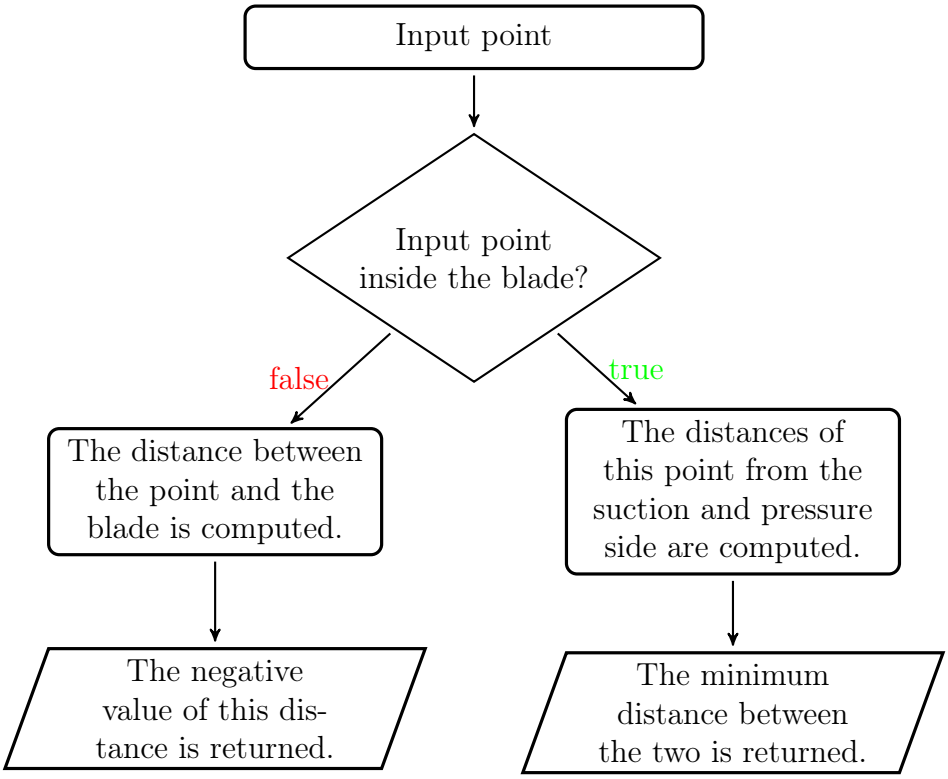


Figure 7.13: Implementation of `constraintValue` function.

- If the point P is inside the volume of the blade, the two distances of the point P from the surfaces of the blade (suction/pressure side) are computed. Then, the minimum distance between these two distances is returned as the constraint value, such that the constraint is satisfied.
- If the point P is outside the volume of the blade, its distance from the blade is computed and the negative value of this distance is returned as the constraint value, such that the constraint is violated.

Finally, as shown in Fig. 7.14, the constraint value for the first cylinder (C_1) is expressed by the system 7.9:

$$C_1 = \begin{cases} d_{in.min}, & \text{if all the points are inside the blade,} \\ -d_{out.max} & \text{otherwise.} \end{cases} \tag{7.9}$$

By using this approach, the function that computes the value of the inequality constraint is `constraintValue`. This means that, while using the *reverse mode* version of the differentiated **OCCT**, `constraintValue` is the unique function that needs to be traced for the computation of the derivative of the constraint.

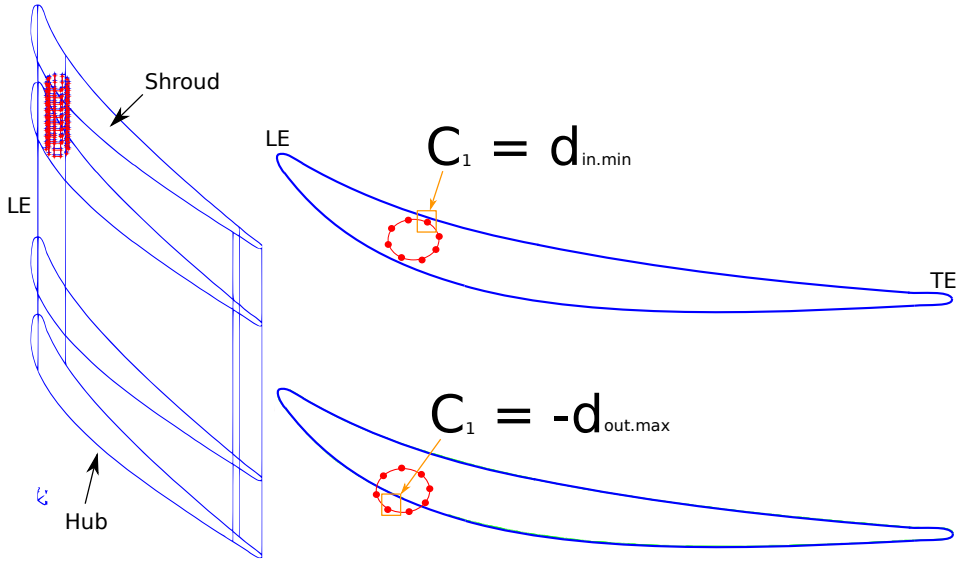


Figure 7.14: Blade skeleton (in this example the blade is characterised by 4 sections) with mesh points for one cylinder, on the left. Example of constraint satisfied (top)/violated (bottom), on the right.

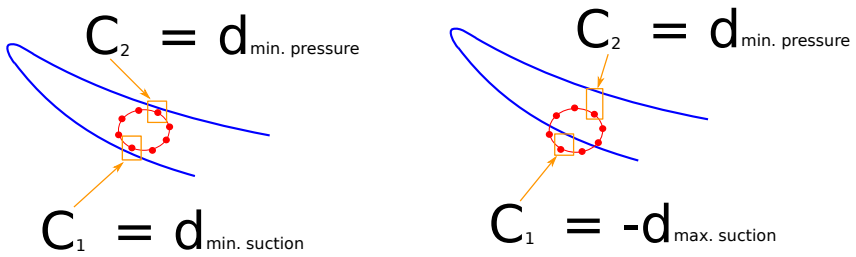


Figure 7.15: Example of computation of two constraints per cylinder. On the left, case with both constraints satisfied. On the right, case with suction constraint violated.

The approach described above does not detect the side of the blade (suction or pressure side) closest to the cylinder, i.e. the distance between the cylinder and each side of the blade is not considered. This information allows the optimiser to consider the margin w.r.t. both sides of the blade during the flow optimisation, i.e. to verify to which side (suction or pressure) the hole is closer. This information can facilitate the convergence of the optimisation.

In order to provide the optimiser with the margin w.r.t. both sides of the blade, two constraints will be imposed: C_1 and C_2. If all the mesh points are inside the

blade, the value of the constraints is computed as in the system 7.10:

$$\begin{cases} C_1 = d_{min.suction}, \\ C_2 = d_{min.pressure}, \end{cases} \tag{7.10}$$

where $d_{min.suction}$ and $d_{min.pressure}$ are the minimal distance between the mesh and the suction and pressure side of the blade, respectively.
Otherwise, if certain mesh points are outside the volume of the blade and, for example, the suction side is violated (as shown in Fig. 7.15), the value of the constraints will be the ones presented in the system 7.11:

$$\begin{cases} C_1 = -d_{max.suction}, \\ C_2 = +d_{min.pressure}, \end{cases} \tag{7.11}$$

where $d_{max.suction}$ is the maximum distance between the part of the mesh that is outside of the blade and the violated side (the suction one, in this case). The second constraint value C_2 is equal to $d_{min.pressure}$, which is the minimum distance between the part of the mesh that is inside the volume of the blade and the "not-violated" side (the pressure side, in this case).
As explained in the following section, this type of approach allows the convergence of the CAD-based aerodynamic shape optimisation of the TUB test case with all the constraints embedded in the flow optimisation.

7.5.2 TUB low fidelity optimisation

The flow optimisation is carried out by using a low fidelity mesh, which consists of 16k cells (Fig. 7.16). The optimiser utilised is the SLSQP. As explained in Subsec. 7.2.2, this optimiser allows to respect all the manufacturing constraints mentioned in Subsec. 6.8.2. The assembly constraints are imposed with the AC4 approach explained in Subsec. 7.5.1 such that a total number of eight inequality constraints is considered ($C_{1...8} \geq 0.$, Fig. 7.16). The hub/shroud cylinders are set at 60 mm distance between each other (i.e. $C_9 - 60 = 0.$ and $C_{10} - 60 = 0.$, Fig. 7.16). To simplify the optimisation, the cylinders are not allowed to move such that the constraints related to C_9 and C_{10} (i.e. $C_9 - 60 \geq 0.$ and $C_{10} - 60 \geq 0.$) have not to be imposed during the flow optimisation.

The optimisation successfully converged after 13 iterations (Fig. 7.17). The simulation is performed by using a coarse mesh, which makes the flow results unreliable. However, this optimisation is relevant because it demonstrates that the

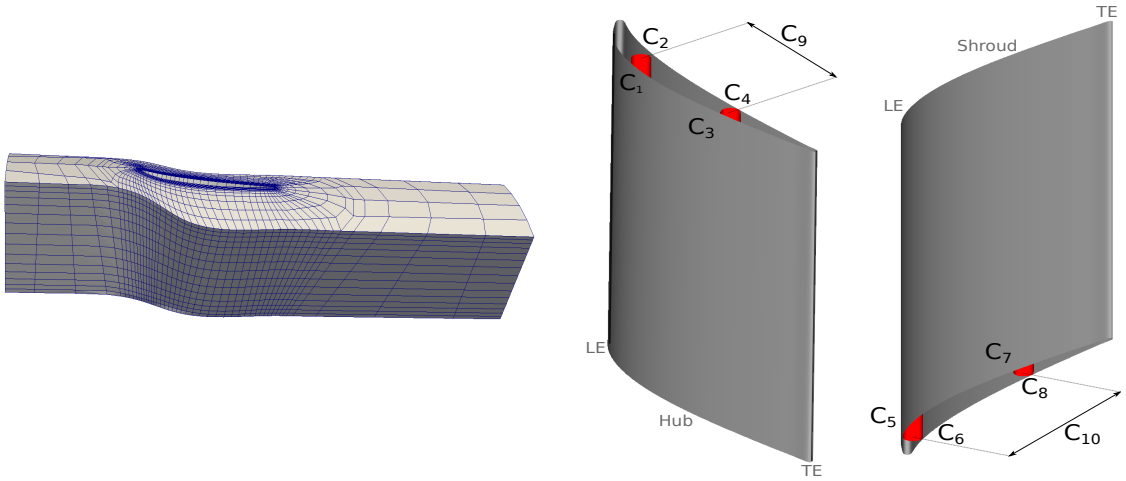


Figure 7.16: Left: low fidelity mesh used to test the AC4 approach to fit the assembly constraints. Right: set of assembly constraints respected during the flow optimisation.

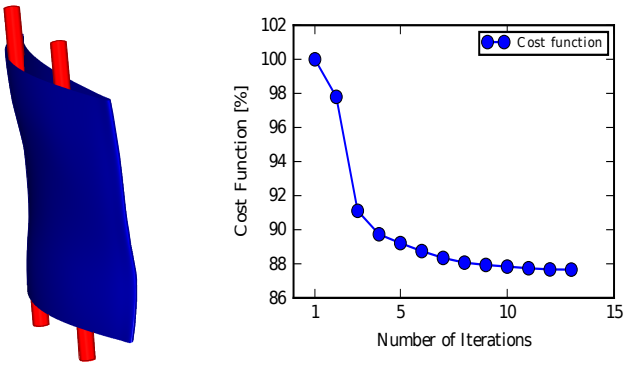


Figure 7.17: Left: optimised blade with cylinders (the cylinders exceed the dimensions of the blade to facilitate their visualisation). Right: optimisation history of the low fidelity optimisation of the TUB by using the AC4 approach to impose the assembly constraints.

$C_{1,in} = 0.42$ mm	$C_{1,opt} = 0.001$ mm
$C_{2,in} = 0.46$ mm	$C_{2,opt} = 0.034$ mm
$C_{3,in} = 1.66$ mm	$C_{3,opt} = 0.01$ mm
$C_{4,in} = 1.68$ mm	$C_{4,opt} = 1.88e^{-5}$ mm
$C_{5,in} = 0.42$ mm	$C_{5,opt} = 0.01$ mm
$C_{6,in} = 0.46$ mm	$C_{6,opt} = 0.62$ mm
$C_{7,in} = 1.65$ mm	$C_{7,opt} = 0.01$ mm
$C_{8,in} = 1.68$ mm	$C_{8,opt} = 0.0001$ mm

Table 7.1: Value of the constraints for the baseline and the optimised geometry. At the end of the optimisation, all the constraints (whose definition is reported in Fig. 7.16) are respected.

AC4 approach can be used to successfully impose the TUB assembly constraints. Table 7.1 reports the baseline and the optimised value of the constraints $C_{1...8}$.

7.6 TUB compressor blade optimisation with all constraints embedded in the optimisation loop

To perform the CAD-based shape optimisation of the TUB test case while embedding the assembly constraints in the flow optimisation, the settings described in the Sec. 6.8 are considered. Therefore, the parametrisation is the one described in Sec. 5.8 while the design chain (reported in Fig. 7.18) has been analysed in Subsec. 6.2.2.
The settings that are changed w.r.t. the simulation presented in Sec. 6.10 are the following. Firstly, the total number of design variables is 188. Indeed, 184 design parameters are the variables used to parametrise the CAD model of the blade. Additional 4 parameters are defined to let the cylinders move during the flow optimisation. In particular, the cylinders are allowed to move along the mid-line M of the top (for the casing cylinders)/bottom (for the hub cylinders) section of the blade, as shown in Fig. 7.19. The mid-line represents the middle path between the suction and pressure side of the 2D section. In this research, it differs from the camber-line, which is the middle path between the two polygons controlling the suction and the pressure side of the 2D section (Sec. 4.2). To define the design parameters that control the centre of the cylinders, the parametric form of the mid-line curve, M = M(p), is used. By using this formulation, any point of the mid-line can be retrieved after the definition of the correspondent parameter p. The (x, y) coordinates of the computed point are then assigned to the (x, y)

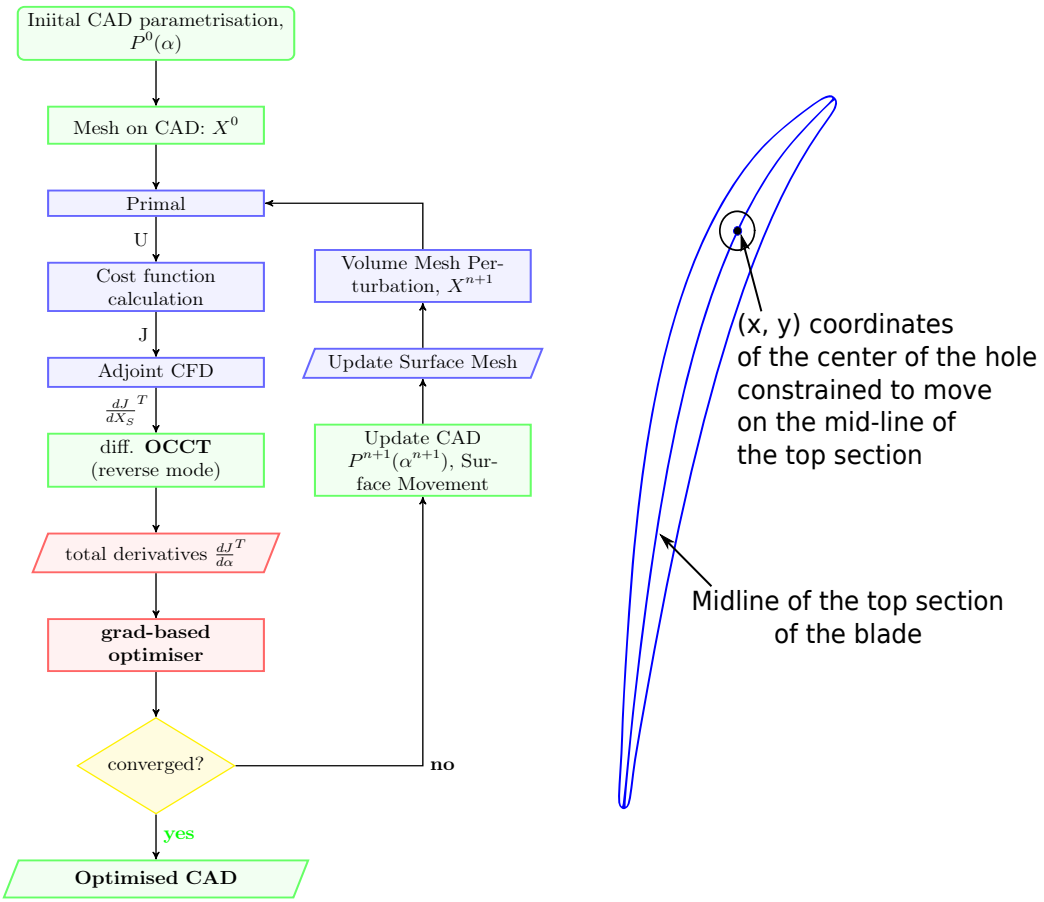


Figure 7.19: Left: optimisation framework. Right: design parameter of the cylinder.

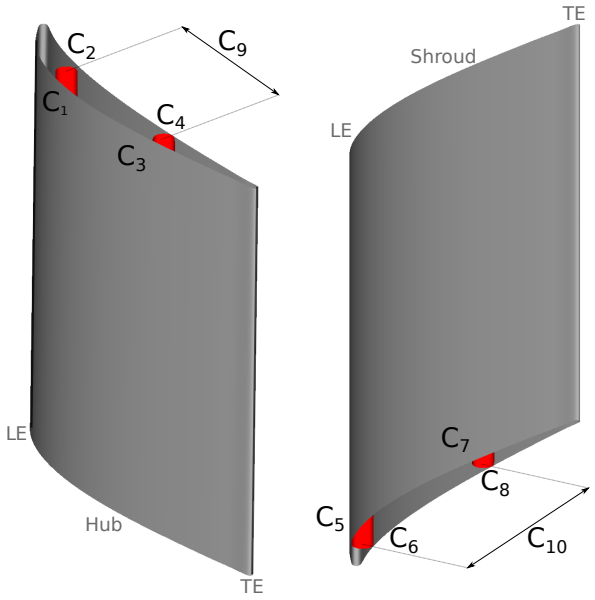


Figure 7.20: Set of assembly constraints respected during the flow optimisation.

coordinates of the centre of the cylinder.
Based on this approach, one additional design parameter (the p parameter of the top/bottom mid-line) is considered per every cylinder. Therefore, four variables are added to the 184 design parameters of the blade such that the complete design space consists of 188 design parameters.
The second main difference with the simulation shown in the previous chapter (Sec. 6.10) relies on the optimiser chosen for the flow optimisation. As mentioned at the beginning of this chapter, the optimiser is the SLSQP from Python *SciPy* library. This optimiser allows to satisfy additional equality/inequality constraints. These constraints are imposed to embed the assembly constraints during the flow optimisation. The total number of inequality constraints is 10 (Fig. 7.20). Eight constraints, which are computed using the approach explained in Sec. 7.5, ensure that four cylinders fit inside the blade.
Moreover, the additional requirement of keeping the minimum axial distance of 60 mm between the cylinders at the shroud and at the hub side also has to be fulfilled. Therefore, two additional inequality constraints, C_9 for the casing bolts and C_{10} for the hub ones, are imposed. These constraints (Fig. 7.20) are shown in the system 7.12:

$$\begin{cases} C_9 = d_{casing\_cylinders} - 60. \geq 0., \\ C_{10} = d_{hub\_cylinders} - 60. \geq 0. \end{cases} \tag{7.12}$$

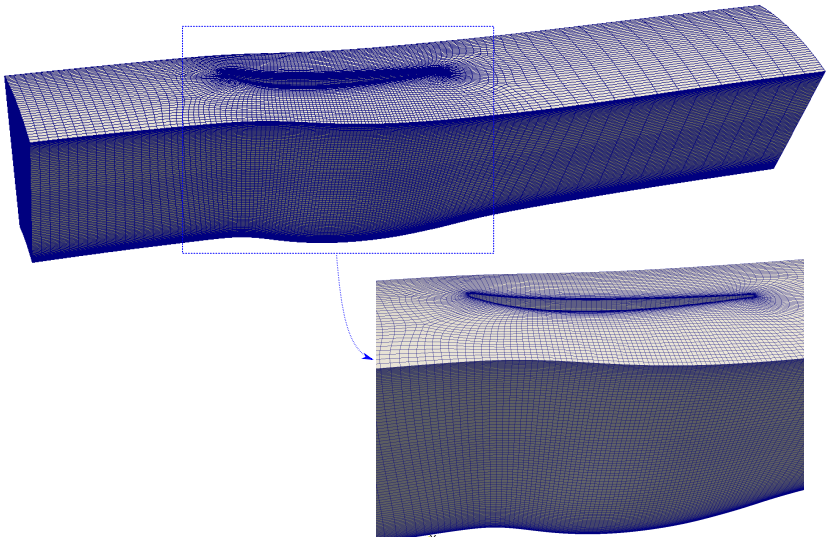


Figure 7.21: The computational grid used to perform the shape optimisation of the TUB test case with embedded the assembly constraints.

where $d_{casing\_cylinders}/d_{hub\_cylinders}$ is the distance between the casing/hub cylinders. Finally, the computational grid (shown in Fig. 7.21) consists of 800k nodes (Sec. 6.9).

The optimisation successfully converged after 43 iterations and the cost function (ΔP between the inlet and the outlet of the stator) is reduced by 14%, as shown in Fig. 7.22. An important remark about the optimisation history is that the cost function does not drop continuously until the end of the optimisation. The reason of this is that in some iterations the drop of the cost function causes the violation of one/several constraints. These violated constraints are restored by the optimiser in the following iterations. The convergence history of all the constraints is shown in Fig. 7.22.

The optimisation result implies the following comments. Firstly, the optimised blade respects all the constraints, as can be verified in Table 7.2 where the value of each inequality constraint is presented at the first and the last iteration of the optimisation. Then, as shown in Fig. 7.23, the optimised blade presents a different thickness distribution, which makes the blade thicker w.r.t. the baseline geometry with a strong camber-line movement at the mid section of the blade (design mode T_1). A thicker blade is affected by a reduced load. This allows to decrease the ΔP between the inlet and the outlet of the stator [157]. Fig. 7.23 shows a second design mode (design mode T_2): the optimised geometry provides a displacement of the trailing edge which allows to reduce the flow separation presented by the baseline geometry at the TE. As a consequence of the two major design modes

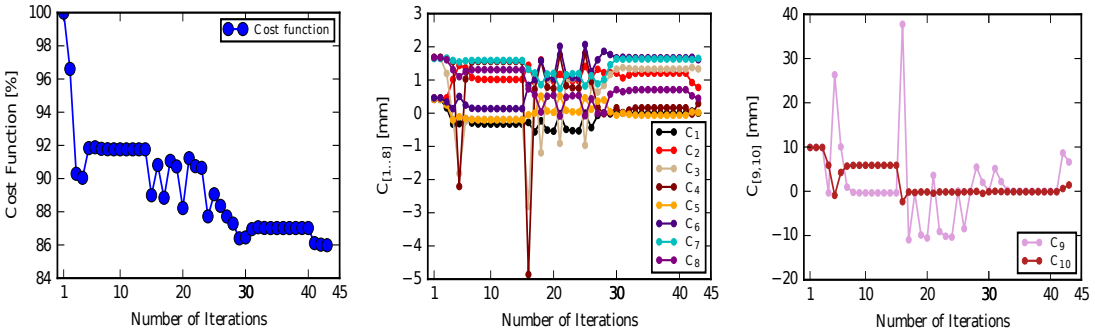


Figure 7.22: Left: convergence history of the cost function. Centre: convergence history of the constraints $C_{1..8}$. Right: convergence history of the constraints $C_{9,10}$.

$C_{1,in} = 0.42$ mm	$C_{1,opt} = 0.0066$ mm
$C_{2,in} = 0.46$ mm	$C_{2,opt} = 0.63$ mm
$C_{3,in} = 1.66$ mm	$C_{3,opt} = 1.33$ mm
$C_{4,in} = 1.68$ mm	$C_{4,opt} = 0.46$ mm
$C_{5,in} = 0.42$ mm	$C_{5,opt} = 0.082$ mm
$C_{6,in} = 0.46$ mm	$C_{6,opt} = 1.52$ mm
$C_{7,in} = 1.65$ mm	$C_{7,opt} = 1.68$ mm
$C_{8,in} = 1.68$ mm	$C_{8,opt} = 0.40$ mm
$C_{9,in} = 9.90$ mm	$C_{9,opt} = 5.97$ mm
$C_{10,in} = 9.90$ mm	$C_{10,opt} = 2.36$ mm

Table 7.2: Constraints values for the baseline and the optimised geometry. At the end of the optimisation, all the constraints (whose definition is reported in Fig.7.20) are respected.

(T_1 and T_2) presented by the optimised geometry, we can verify an increase of the outlet angle of the flow w.r.t. the baseline geometry (Fig. 7.24). This result was expected because the cost function is the ΔP between the inlet and the outlet of the stator. This cost function can be extensively decreased by unloading the blade, i.e. by changing the exit angle of the blade. This behaviour is limited in this simulation by fixing the top and bottom slices of the blade during the flow optimisation and by defining a minimum thickness distribution to be respected (Subsec. 6.8.2).

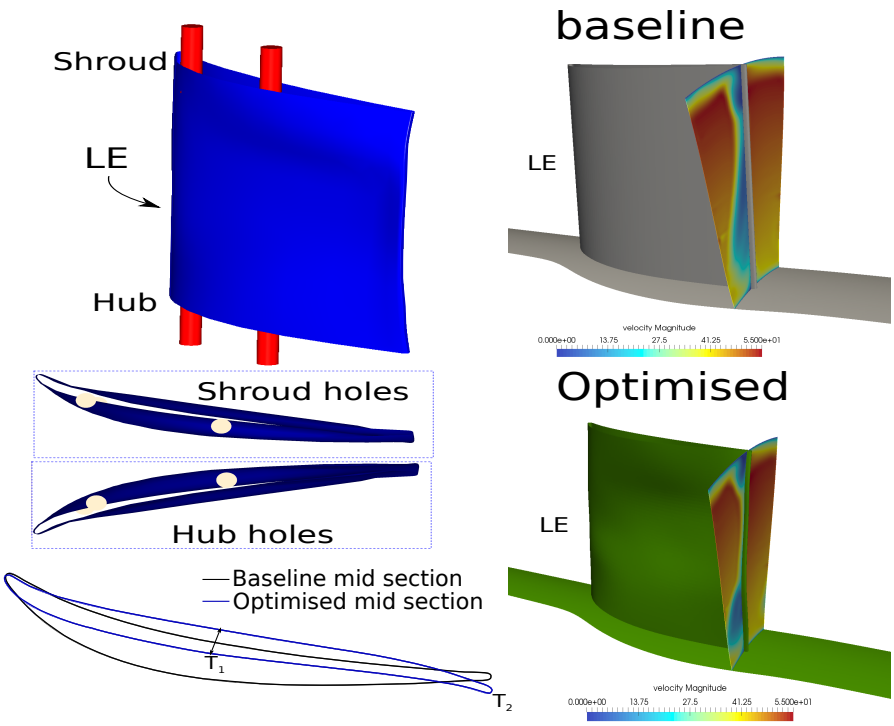


Figure 7.23: Left, top: optimised blade with the cylinders inserted in the volume of the blade. The cylinders are exceeding the volume of the blade just to facilitate their visualisation. Left, middle: position of the optimised cylinders at the casing and the hub. Left, bottom: mid section of the baseline and optimised geometry. In this part of this figure the main design modes T_1 and T_2 are shown. Right: comparison between the velocity magnitude of the initial (top) and final (bottom) iteration of the optimisation at the TE.

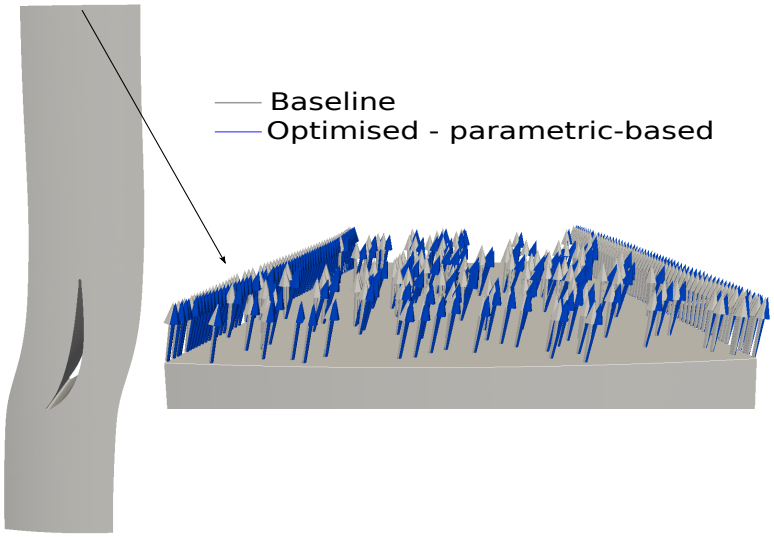


Figure 7.24: Outlet of the stator: the optimised blade increases the flow exit angle w.r.t. the baseline geometry.

7.7 TUB: comparison with reference results

7.7.1 Reference results

In this thesis, the author has defined as reference results the optimal shapes provided by parametrisations with refined design space. For the TUB test case, it is not possible to find in the literature optimised shapes that could be considered as reference because the fit of the assembly constraints during the gradient-based optimisation has never been demonstrated for the TUB test-case. Therefore, the author has obtained the reference results by developing a parametric-based CAD model of the blade with a design space enlarged w.r.t. the parametrisation utilised in Sec. 7.6 (i.e. w.r.t. the parametric-based design).
The refined parametrisation (Fig. 7.25), which consists of 656 design parameters, is obtained from the parametric-based design (184 design parameters) as follows:

- the number of parameters of the 2D section is increased from 23 to 41. The design parameters of the leading and trailing edge are fixed (radius of curvature and (x, y) coordinates, Fig. 7.25). The remaining design parameters are doubled. The points distributed on the camber-line to control the thickness distribution are increased from 8 to 16. The control points of the camber-line raise from 5 to 10. Therefore, the total number of design parameters that control the 2D section is 41: 18 parameters for the thickness distribution (radius of LE, radius of TE and other 16 thickness parameters)

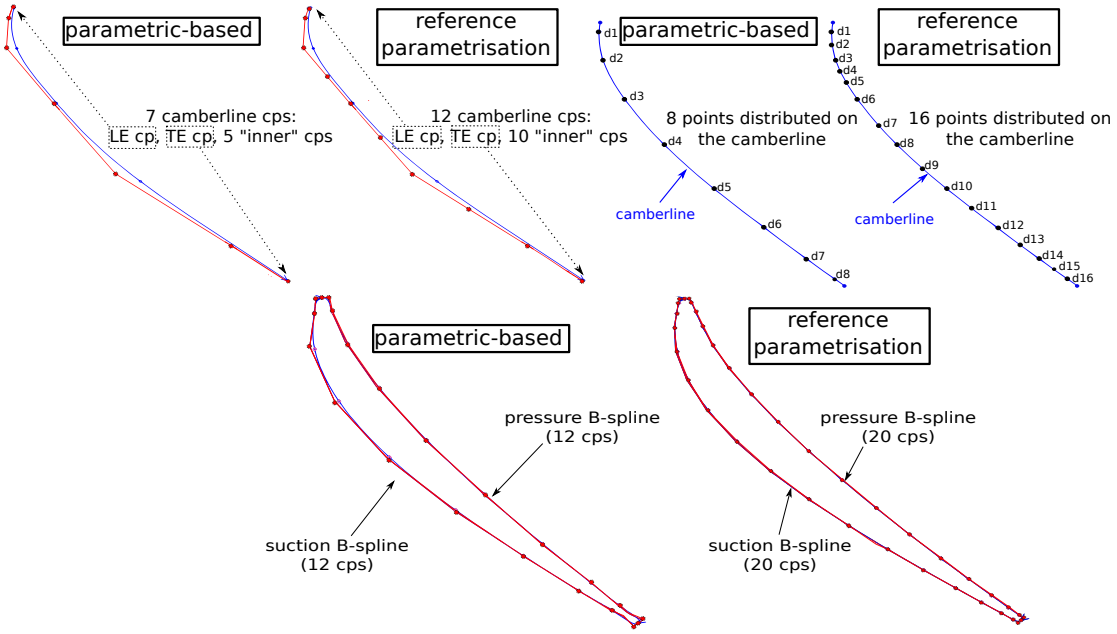


Figure 7.25: Comparison: parametric-based vs the refined parametrisation. Top, left: the camberline of the refined parametrisation consists of 12 control points (cps). Top, right: the number of points distributed onto the camberline for the refined parametrisation are 16. Bottom: the suction and the pressure sides of the refined parametrisation consists of 20 cps.

and 23 parameters for controlling the camber-line (Fig. 7.25). The author highlights that the design parameters of the camberline are reduced from 24 ((x, y) coordinates of LE, TE and 10 control points) to 23 because the y-coordinate of the TE is defined as the y-coordinate of the LE summed to the constant axial chord value. Fig. 7.25 reports the 2D section set as reference parametrisation. This parametrisation is successfully refitted to the baseline geometry of the TUB 2D section by solving a fitting problem with the same settings explained in Sec. 4.3 for the medium parametrisation.

- The number of control points per law of evolution is doubled (from 8 to 16).

Totally, the design space of the shape optimisation performed with the enlaged design space consists of 660 design parameters because the four design parameters necessary to consider also the movement of the cylinders during the flow optimisation are added to the 656 design parameters of the blade. Apart from the design space, the same settings of the optimisation shown in Sec. 7.6 are used. The optimisation successfully converged after 33 iterations. The optimised blade reduces the cost function by 16.8%, as shown in Fig. 7.26 where the history of the optimisation is reported.

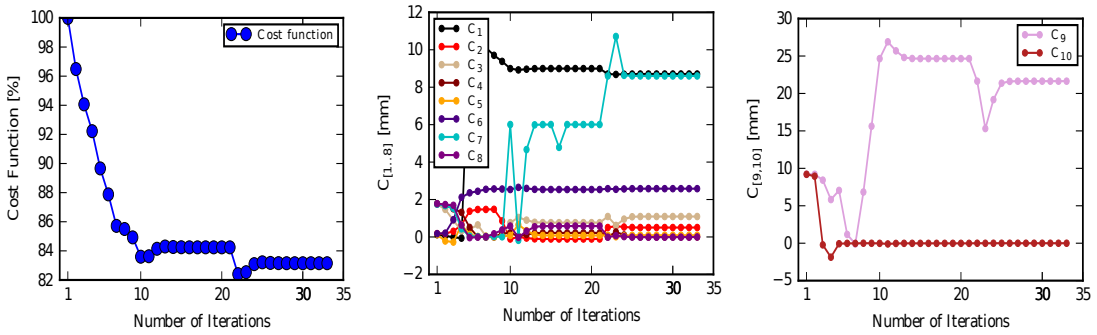


Figure 7.26: Optimisation history of TUB test case with enlarged design space. Left: convergence history of the cost function. Centre: convergence history of the constraints $C_{1..8}$. Right: convergence history of the constraints $C_{9,10}$.

7.7.2 Parametric-based vs reference results

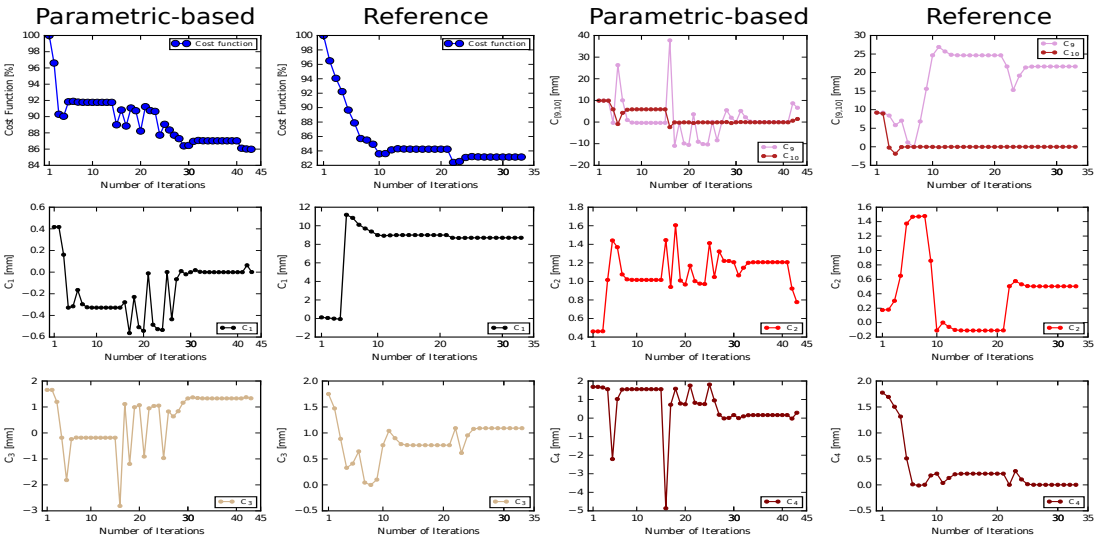


Figure 7.27: TUB:comparison between optimisation histories of reference and parametric-based results (part 1).

In this section, the comparison between the results obtained with the parametric-based design and the optimal shape given by the refined parametrisation is presented.

Firstly, an increased design space converges faster (i.e. with a reduced number of iterations) w.r.t. the parametric-based design. Normally, this is not guaranteed when performing shape optimisations. A too rich design space could cause an increase in the number of iterations necessary to reach the convergence. On the other hand, when executing the shape optimisation with the parametric-based

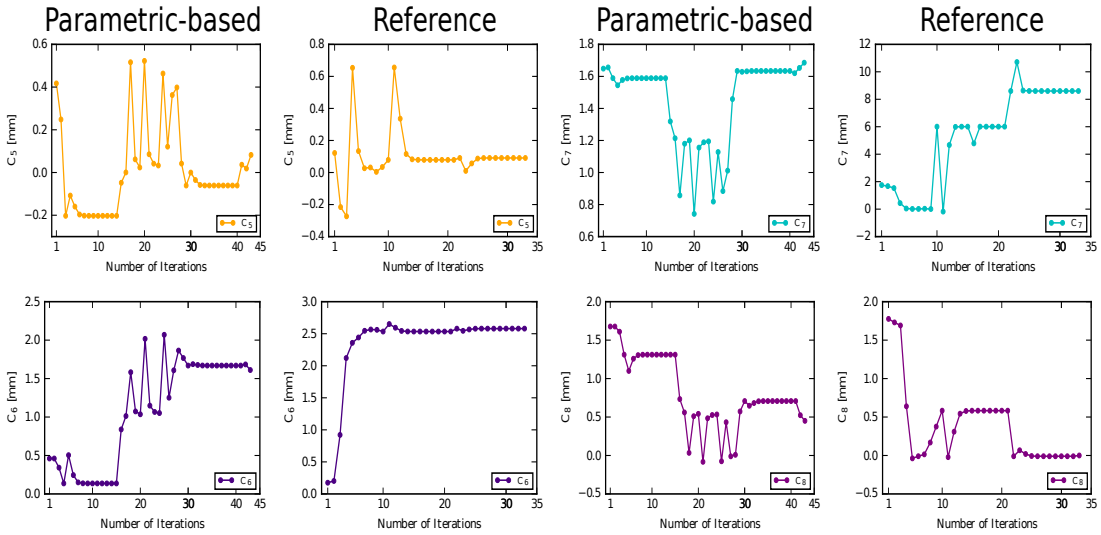


Figure 7.28: TUB:comparison between optimisation histories of reference and parametric-based results (part 2).

design (the design space consists of 184 design parameters), the optimiser violates the constraints. This slows down the convergence of the optimisation because the optimiser, after a constraint has been violated, needs some iterations to recover the violated constraint (Fig. 7.27, Fig. 7.28). As the plots of the constraints in Fig. 7.27 and Fig. 7.28 show, the constraints of the increased design space are rarely violated during the flow optimisation. This facilitates the convergence of the optimisation. The author highlights that the value of the constraints C_1, C_2, C_5, C_6 at iteration 1 differs between the parametric-based design and the reference parametrisation. This difference is due to the fact that the two parametrisations have been refitted to the baseline 2D section of the TUB by solving two different fitting problems. The settings of both fitting algorithms are the same (Sec. 4.3) but the different design spaces considered during the optimisation (23 design parameters vs 41 design parameters) has lead to minor differences between the two refitted shapes. This causes a change also in the evaluation of the value of the constraints C_1, C_2, C_5, C_6.

Secondly, the reference shape picks up the same design modes T_1 and T_2 considered by the parametric-based design (Sec. 7.6). Despite this, the reference parametrisation provides a better reduction of the cost function (16.8% vs 14%, Fig. 7.27). The margin (2.8%) w.r.t. the results provided by the parametric-based design (Fig. 7.22) is related to the fact that the reference parametrisation gives an optimal shape which presents a stronger displacement of the TE (design mode T_2, Fig. 7.29). As shown in Fig. 7.29, this allows to further reduce the tip flow

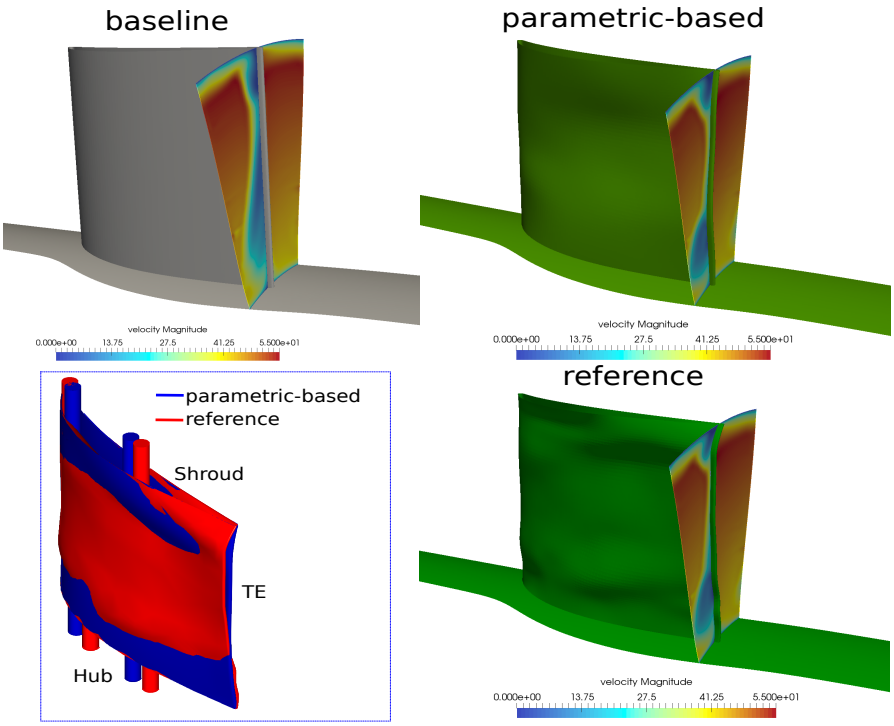


Figure 7.29: Top, left: flow at the TE for the baseline geometry. Top, right: flow at the TE for the optimised shape with the parametric-based design . Bottom, right: flow at the TE for the reference. Bottom, left: comparison between the optimised shape provided by the parametric-based design and the reference one.

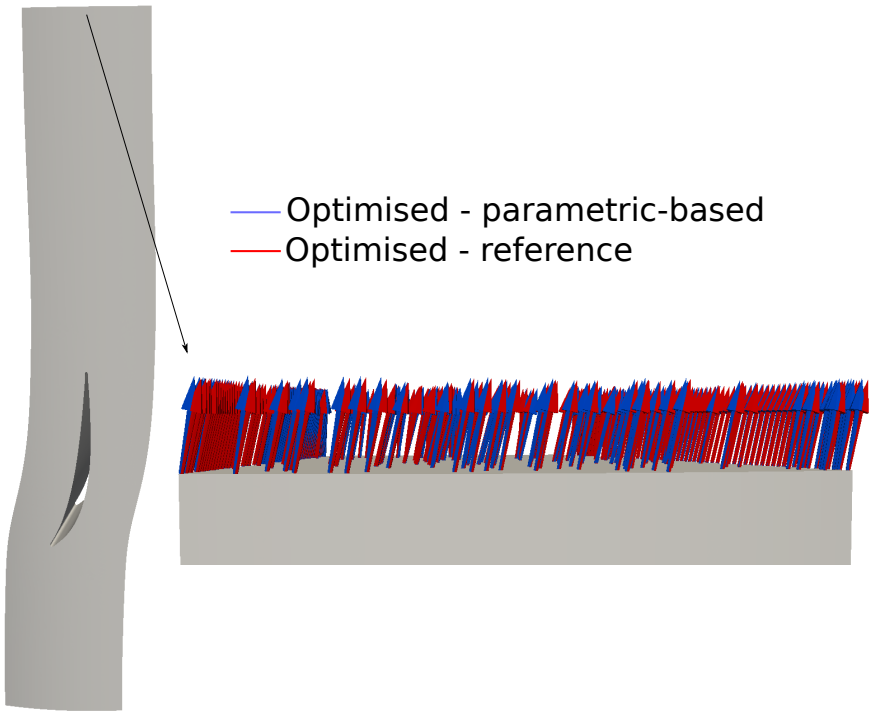


Figure 7.30: TUB, flow exit angle: comparison between parametric-based and reference results.

separation. As a consequence of the different shape at the trailing edge, also the exit angle of the flow given by the reference results increases w.r.t. the parametric-based ones (Fig. 7.30).
The optimised position of the bolts is different. This is mainly related to the higher refinement considered by the reference parametrisation, which allows to better control locally the shape and to accommodate the bolts differently w.r.t. the parametric-based design proposed in this thesis.
Finally, the author highlights that another possible cost function to be considered is the weighted sum of two terms: (i) the Δp between the inlet and the outlet of the blade and (ii) the flow exit angle. The minimisation of one of these two terms can be achieved by increasing the other one, such that the goal of the flow optimisation would be to identify the shape which provides the best compromise between these two effects. This new cost function is one of the next steps of this research.

7.8 Imposition of the assembly constraints: discussion

The successful imposition of the assembly constraints for the optimisation of the TUB compressor stator blade is an important step which has never been demonstrated before [75, 152, 157]. The respect of such constraints allows to straightforwardly reintroduce the optimised geometry into the CAD model of the final product (Sec. 5.1). This facilitates the application of gradient-based design chains to industrial problems.

As shown in Sec. 7.6, the signed distance approach AC4 (Sec. 7.5) allows to identify a differentiable assembly constraints function. This approach is enticing for the scientific community because it is generic, i.e. it can be extended to other type of assembly constraints. For example, the AC4 approach can be applied to fit fastener such as screw joints. Also, one could discretise with a point cloud the geometry which defines the neighbour components and perform the shape optimisation by imposing that the blade never exceed the space defined by such point cloud. This would guarantee that the optimised blade surely fits into the CAD model of the final product.

The author highlights that, for the TUB test case, other approaches, which are

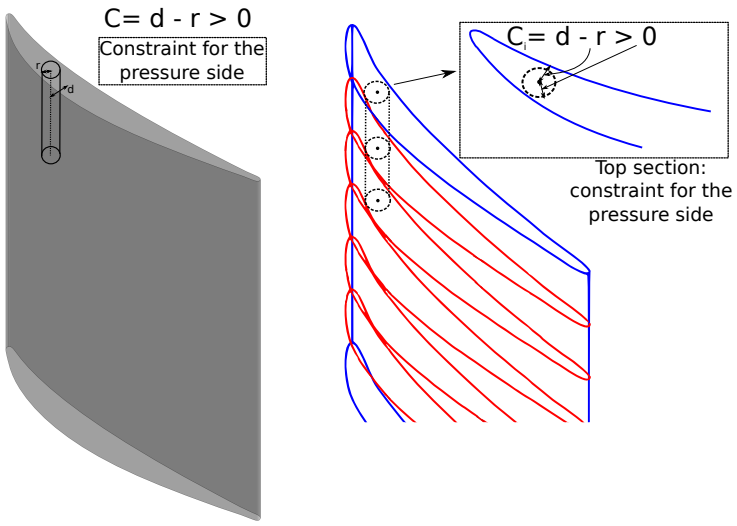


Figure 7.31: Other possible approaches to impose the assembly constraints. On the left, for each side of the blade a constraint is imposed (i.e. the minimum distance between the axis of the cylinder and the closest point on the suction/pressure side is bigger then the radius of the cylinder). On the right, a set of constraints is computed (one for every section of the blade) for every side of the blade.

tailored to this specific assembly constraint (i.e. the cylinder where the bolt have to be accommodated), could also be investigated. A first approach is to impose that the minimum distance between the axis passing through the centre of the cylinder and the suction/pressure side of the blade is always bigger then the radius of the cylinder such that the intersection between the blade and the assembly constraint never occurs (Fig. 7.31). Similarly, one could consider a constraint which computes for every 2D section the minimum distance between the centre of the cylinder and the suction/pressure curve. The 2D constraint sets as positive the difference between this minimum distance and the radius of the cylinder, as shown in Fig. 7.31.
The aforementioned approaches differ from the AC4 one for the formula (i.e. algorithm) used to compute the signed distance. The algorithm which provides the calculation of the minimal distance between a line (i.e. the axis passing through the centre of the cylinder) and a surface (e.g. suction side) or between a point (i.e. the centre of the cylinder) and a curve can be implemented by using the subroutines `GeomAPI_ProjectPointOnSurf` and `GeomAPI_ProjectPointOnCurve`, which compute the projection of a point onto a surface and onto a curve, respectively. Such subroutines do not always calculate correctly the position of the projected point [158]. This does not make these approaches straightforward to utilise. The investigation of these two approaches requires further research work.

7.9 Summary

This chapter has presented an investigation about the possible approaches that can be used to embed the assembly constraints during the flow optimisation of the TUB compressor stator blade. The utilisation of the intersection algorithms given by OCCT to define the function of the assembly constraint has been demonstrated to be not reliable. This is mainly due to the difficulties in identifying a differentiable primal.
The alternative solution proposed by the author has been to compute the function of the assembly constraint as signed distance to the component to be optimised (i.e. the blade) of sampled points on the assembly constraint (i.e. the cylinder). The simplicity of the algorithm that implements such function has allowed to define a differentiable primal, which has been used to embed the assembly constraints during the aerodynamic shape optimisation of the TUB.
The results obtained by using the parametric-based design proposed in Sec. 5.8 (which consists of 184 design parameters) have been compared to the optimal

shape provided by the reference parametrisation (656 design parameters, Subsec. 7.7.1). These two parametrisations provide a similar reduction of the cost function. Also, a general agreement is verified between the geometric features of the two optimised geometries. These geometries pick up the same design modes T_1 and T_2 (which affect the thickness distribution and the TE position) in order to minimise the cost function.

Chapter 8

Conclusions and further work

8.1 Summary

The utilisation of shape optimisation is spreading in industry to improve the aerodynamic performance of the product. The computational costs of stochastic or meta-modelling optimisation methods scale with the number of design variables such that hundreds or thousands flow evaluations can be required to converge. If the number of design variables is sufficiently high (e.g. bigger then 100), gradient-based methods need less optimisation iterations to reach the convergence as a mono-dimensional path through the design space has to be traced.

Gradient-based methods require the calculation of the total derivatives, i.e. the derivatives of the cost function w.r.t. the design parameters that control the shape. These derivatives can be split into two terms: (i) the CFD derivatives, which can be computed with the efficient adjoint CFD method and (ii) the shape derivatives, i.e. the derivatives of the surface mesh points w.r.t. the design parameters.

The shape derivatives can be calculated either with CAD-free parametrisations or with CAD-based ones. CAD-free parametrisations allow to straightforwardly compute these derivatives. Their main limitation is that they do not parametrise the geometry with CAD models. The CAD model is used in industry to share the geometry with various tools in the design, analysis and production workflow.

CAD-based parametrisations do parametrise the geometry with CAD models. The utilisation of CAD models in shape optimisation is hindered by the difficulties to compute the shape derivatives. These derivatives can be calculated with several approaches: FD, complex variables or hand-differentiation. If the sources of the

software are available, algorithmic differentiation is an enticing solution since, by employing the reverse mode, AD exactly and efficiently compute the derivatives. In this research, the algorithmically differentiated version of the open source CAD library OCCT is utilised to parametrise the geometry.

OCCT is one of the main CAD kernels available on the market and is the only one open source. It consists of thousands of C++ classes and provides solutions in the areas of surface and solid modelling, 3D and 2D visualisation and data exchange. OCCT is differentiated by using the AD tool ADOL-C. Both *reverse* and *forward mode* are successfully implemented. The implementation of these modes has been conducted by one of the collaborators of the EC project IODA, Mladen Banovic.

This thesis mainly focuses on the development of automatically differentiated parametric CAD models to perform aerodynamic shape optimisation. The main advantages of utilising these CAD models in a gradient-based shape optimisation framework are investigated and demonstrated.

In engineering design, the implementation of such CAD models is normally achieved by using the parametric features provided by commercial CAD systems such as CATIA. This implementation is obtained in this thesis by using the C++ classes provided by OCCT. This requires extensive efforts spent in code development because all the CAD algorithms relevant to perform the shape optimisation are coded manually. Among these algorithms, the main ones are: the hierarchic feature tree to implement the CAD models, which allows to control the geometry with the parameters normally used by the designer in the conception phase of the product, the explorer which extracts the list of subshapes (e.g. curves) that compose a datum CAD model, the subroutine which implements the properties of shapes (e.g. computation of the medium line of a given 2D profile) and the algorithms to classify a point in a solid (e.g. check if a point is inside a volume).

The test cases studied in this thesis are two turbomachinery components: a cooling channel for high-pressure turbine blades (the U-bend) and a compressor stator blade. These test cases have been identified within the European Commission framework IODA. They are realistic components which allow to study two aerodynamic shape optimisation problems typically investigated in industry. The U-bend prescribes the fluid-dynamics optimisation which aims to avoid the vor-

texes due to the high curvature of the bend at the U-turn. The compressor stator blade has the goal to minimise the flow separation at the outlet of the stator which is a consequence of the deviation of the flow from the 42° at the inlet of the stator to the 0° at the outlet (i.e. the axial direction).

For both test cases, the parametrisation relies on a cross-section design approach. The CAD models are implemented by approximating with a single B-spline surface a set of 2D cross-sections constructed along a guiding curve. Each cross-section is controlled by a set of B-Spline curves properly arranged to define typical engineering parameters such as camberline and radii at leading and trailing edge. Every parameter evolves along the guiding curve based on a law, which is a B-spline curve. The control points of the laws of evolution are the design parameters of the optimisation.

For both test cases, the derivatives calculated with the CAD models automatically differentiated in forward mode are compared to the derivatives computed with the FD method. This comparison, which can not be used to rigorously validate the derivatives computed with the AD, has allowed to verify the general agreement between the AD derivatives and the FD derivatives. The derivatives calculated with the reverse mode implementation are compared to the derivatives provided by the forward mode. The differences between these two modes of AD is below machine precision.

The baseline geometry of the two test cases is available as STEP file. The parameters of the CAD model of the U-bend are edited manually in order to match the existent baseline geometry. For the TUB, this is not possible because the parameters controlling the geometry such as thickness distribution and the position of the control points of the camberline are not provided together with the STEP file of the test case. This is common to several engineering applications. Therefore, the development of a tool that reparametrises the 2D section of the TUB is necessary. This tool extracts the 2D section of the TUB out of the STEP file by using the explorer which extracts the list of subshapes (e.g. curves) out of a CAD model (mentioned above). Then, the value of the parameters which control the automatically differentiated CAD model (the proposed parametrisation) are computed by solving a gradient-based fitting problem. The cost function to be minimised (Eq. 4.7) is the "total distance" between the proposed parametrisation and the baseline geometry. The design parameters of the optimisation are the

variables that control the automatically differentiated CAD model. The derivatives of the cost function w.r.t. these variables are computed with the efficient AD reverse mode. The optimisation problem successfully converges such that the cost function is almost nullified (the cost function drops by 99.9%).

STAMPS is the CFD/adjoint CFD solver utilised in this research work. The adjoint CFD solver has been retrieved by other researchers at QMUL CFD optimisation group by applying the AD tool Tapenade to the sources of the STAMPS' CFD solver. The CFD derivatives provided by this adjoint CFD solver are plugged as seed vector into the differentiated OCCT (reverse mode), which then computes the total derivatives. The total derivatives are given to an optimiser, which calculates the set of design parameters that generate the shape with improved performance.

In Ch. 6, this fully differentiated design chain is used to perform the shape optimisation of the U-bend (cost function: ΔP between the inlet and the outlet of the bend). The baseline geometry presents a significant flow separation after the U-turn, which continues in the downstream leg. The U-turn is the part of the geometry that needs to be optimised while the inlet and the outlet legs do not change during the flow optimisation. The parametric CAD model imposes the G1 continuity constraint between the U-turn and the inlet and outlet legs. The optimised CAD model substantially reduces the flow separation after the U-turn. This optimised CAD model is compared to the optimised shape provided by the reference parametrisation, which is identified in literature. Despite some differences, the two optimised shapes pick up the same design modes. This verifies that the parametrisation approach proposed in this thesis allows to reproduce the geometric features which optimise the cost function.

The TUB test case is strongly influenced by its manufacturing constraints. These constraints are imposed within the CAD model, with the exception of the assembly ones. Ch. 7 investigates four possible approaches which can be used to fit the assembly constraints (four mounting bolts) during the aerodynamic shape optimisation. The most promising approaches are AC1, AC2 and AC3, which rely on the intersection algorithms implemented within OCCT. By using all these approaches, it would be possible to impose the assembly constraint also for complex geometries with several points of contact/minimal distances, such as a duct in a complex engine bay. The AC4 approach samples a set of points onto the assembly

constraint and compute a signed distance between the assembly constraint and the blade.

The approaches AC1, AC2 and AC3 are tested in geometric optimisation problems. The first optimisation problem considers a blade aerodynamically optimised without considering the assembly constraints such that it is not possible to insert the bolt inside the volume occupied by the blade without occurring in an intersection between the blade and the bolt. The optimisation problem requires the identification of the (x, y) coordinates of the centre of the bolt which allow to minimise this area of intersection. By computing the area of intersection with the AC1 and the AC2 approaches, the optimisation does not converge. The reason of this failure is due to the primal, which is incorrectly computed for some (x, y) coordinates of the centre of the bolt. The erroneous calculation of the area of intersection causes the wrong calculation of the derivatives of the cost function w.r.t. the coordinates of the centre of the cylinder. By using the AC3 approach, which does not provide such erroneous values of the area of intersection, the optimisation converges.

The second optimisation problem, which serves as preparation step towards the aerodynamic shape optimisation, is a surface fitting problem. Two blades are considered: the original blade, controlled by the 184 design parameters defined in this thesis, and the target one, which intersects the bolt at the suction and the pressure sides. The goal of the optimisation is to minimise the total distance between the original blade and the target while respecting the constraint which imposes that no intersection has to occur between the original blade and the bolt. The design parameters of the optimisation are the 184 design parameters controlling the original blade whereas the AC3 approach is used to compute the constraint (i.e. the intersection between the blade and the hole). The optimiser used is the SLSQP implemented within the SciPy library. This geometric optimisation problem does not converge, i.e. the cost function is not minimised and the constraint is still violated after 90 iterations. In order to debug such failure, it would be necessary to investigate how the SLSQP optimiser is implemented within the SciPy library. Unfortunately, this investigation is difficult and requires an extensive amount of time, which has not been available in this research. As a consequence of this failure, the AC3 approach has not been considered as a reliable solution to impose the TUB assembly constraints.

The AC4 approach is successfully used to respect the assembly constraints. All the manufacturing constraints are therefore respected during the flow optimisation. In particular, the best position of the mounting bolts is determined while performing the shape optimisation. The optimal shape given by the proposed parametrisation (the medium size parametrisation) is compared with the reference results, which are obtained by optimising the blade with a refined design space. Both optimised shapes pick up the same design modes, which allow to reduce the cost function (the Δp between the inlet and the outlet of the stator) by increasing the exit angle of the flow w.r.t. the baseline geometry. This behaviour is partially restricted by the additional condition considered in this thesis which fixes the blade at the hub and the shroud.

This research demonstrates the applicability of a fully differentiated design chain in aerodynamic shape optimisation. The CAD features implemented within the CAD kernel OCCT are used to successfully implement the parametrisation. This allows to control the CAD model with the parameters usually handled by the designer during the conception phase and the all manufacturing constraints can be respected during the flow optimisation.

In particular, the fit of the assembly constraints is essential to spread the gradient-based optimisation methods in industry. An optimised geometry that respects the assembly constraints can be immediately reinserted in the overall CAD model of the product. On the contrary, an optimised shape which does not consider the assembly constraints requires further analyses to fit these constraints. The optimised shape should be modified after having performed the shape optimisation. As a consequence, other simulations (i.e. CFD analyses) would be necessary to verify the performance of the modified shape such that the design process is finally slowed down.

The main limitation of the parametrisation approach proposed in this thesis is that the automatically differentiated CAD models are developed by hand. This hinders the spread of such approach in R&D department of industries, where normally designers are not software developers. A possible solution is to define the automatically differentiated CAD model within a CAD system, which is typically used by engineers to define the geometry. The steps necessary to achieve such goal are explained in Sec. 8.4.

8.2 Contributions

This section lists the main contributions of this thesis, which are:

1. designed parametrisations for gradient-based aerodynamic shape optimisation. This includes:
    - investigated the modeling algorithms provided by OCCT and identified the ones that suit shape optimisation.
    - Implemented the automatically differentiated parametric CAD models by using these algorithms.
2. Implemented a reparametrisation tool to refit the existent baseline geometry of the test cases. The derivatives are efficiently computed by using OCCT differentiated in reverse mode.
3. Successfully applied the fully differentiated design chain to optimise the parametric CAD models of the two test cases. This includes:
    - optimised the geometries of the two test cases by using the design space defined within the automatically differentiated CAD models (parametric-based designs).
    - Implemented (for the TUB) the reference parametrisation, which is controlled by a refined design space.
    - Compared the optimised shapes provided by the parametric-based designs w.r.t. the reference results.
4. Imposed the manufacturing constraints. The types of constraint imposed are two:
    - constraints imposed within the CAD model by construction (e.g. tangential continuity between the U-part of the U-bend and the inlet and outlet leg).
    - Constraints imposed by defining boundaries to the design variables within which identify the best set of parameters (e.g. minimum/maximum radius at the TE/LE edge of the compressor stator blade).
5. Investigated OCCT to determine an approach to fit the assembly constraints during the flow optimisation. This includes:

- investigated the intersection algorithms provided by OCCT. These algorithms can not be applied in gradient-based optimisation problems because none of them provides a differentiable primal (i.e. area of intersection function).
- Identified an approach which computes a signed distance between the assembly constraint (i.e. the bolt) and the component to be optimised (i.e. the blade) to fit the assembly constraints during the flow optimisation.

8.3 Next steps of the research

The next immediate steps of this research are:

1. Differentiate a complete CAD system. In this thesis, the CAD models are implemented by using the classes provided by the differentiated OCCT. This could hinder the spread of this approach in industry because usually the designers have not the competences to develop C++ functions. Moreover, the implementation of these functions is time-consuming and error-prone. The solution is to directly parametrise the geometry within a CAD system such that the utilisation of an interactive GUI could facilitate this task. In Sec. 8.4, the author outlines the main steps necessary to differentiate a CAD system based on OCCT: *Shaper*[1].

2. Exploit the structure of the parametrisations when using the differentiated OCCT in *reverse mode*. In this research work, the C++ functions that implement the CAD models have been entirely traced. This implies that the parts of the code which are not used for the calculation of the derivatives are also taped. The total amount of memory occupied is therefore higher then the one really needed to compute the derivatives. For example, an interesting investigation would be to trace only one section of the CAD model and to reuse this trace for the other sections. This would reduce the amount of memory used.

3. Investigate further the intersection algorithms provided by OCCT. In particular, the AC3 approach, which uses the 2D intersection algorithm, should be applied to other optimisation problems in order to identify the reasons of its failure in the geometric application II (Sec. 7.4).

[1] `https://docs.salome-platform.org/latest/gui/SHAPER/index.html`

4. Identify an approach that allows to automatically adjust the design space during the flow optimisation. For example, one could increase the number of control points per laws of evolution based on an analysis of the total derivatives, i.e. if the total derivative of a design parameter exceeds a certain threshold, the number of control points of the law of this design parameter are automatically increased. This would allow to adapt the design space to the specific flow conditions such that the cost function could be further optimised.

5. Couple the differentiated OCCT with other CFD solvers. STAMPS efficiently computes the CFD derivatives but it is still not mature enough to cover all types of test cases. For example, the SA is the only turbulence model implemented. The main open source CFD solvers such as Open FOAM and SU2 do implement also other turbulence models (e.g. K-ϵ, K-ω..). These turbulence models could be used to perform the shape optimisation of test cases which require models different from the SA.

6. Apply the fully differentiated design chain to different problems w.r.t. the ones studied in this research. One of the main goal of this thesis is to demonstrate the applicability of this fully differentiated design chain to aerodynamically optimise parametric CAD models. From the fluid-dynamic point of view, there could be several other studies that could be investigated. It would be interesting to perform optimisations that consider also the movement of the boundaries (e.g.compressor/turbine rotor or the head of the piston of a volumetric machine), the evaluation of unsteady problems, multi-phases calculations.

7. Couple the differentiated OCCT with structural solvers and perform structural analyses. Another interesting study would be to investigate multidisciplinary optimisation by using OCCT as geometric kernel.

8. Manufacture and test empirically the real performance of the optimised geometry. Despite the costs of such activity (which could be reduced by utilising innovative techniques for manufacturing such as 3D printing), this is important to validate the results provided by the fully differentiated design chain.

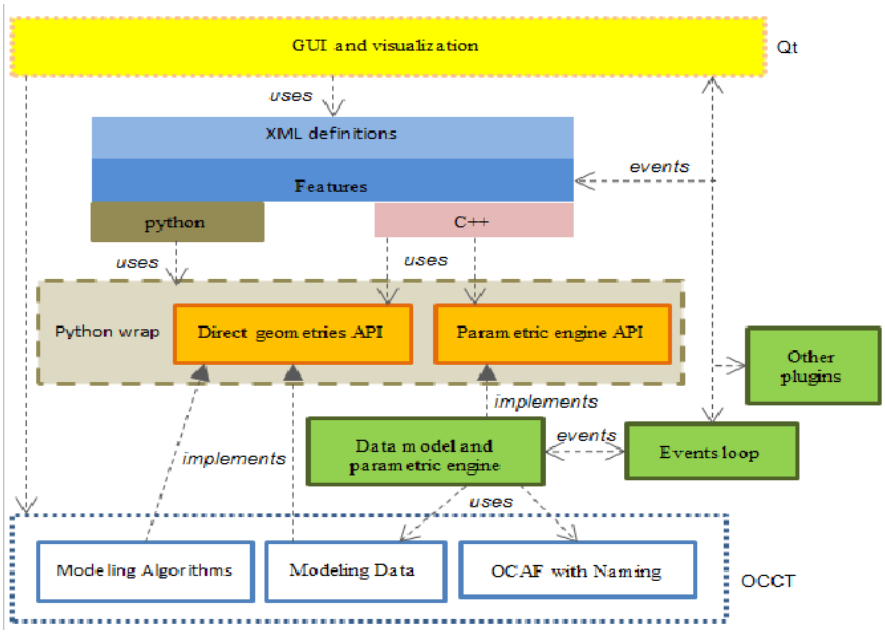


Figure 8.1: *Shaper*'s general architecture [159].

8.4 Differentiation of Shaper: an outline

Shaper is the CAD system developed by *OCC*, EDF and CEA. It is a module inside the pre-/post- processing platform *Salomé*[2]. The utilisation of such CAD system would allow to avoid the main limitation of the parametrisation approach used in this thesis, which is to develop automatically differentiated parametric CAD models by hand. This is time consuming and error-prone (Sec. 5.10).
The structure of Shaper is presented in Subsec. 8.4.1. Subsec. 8.4.2 gives a possible solution to add *Shaper* to the computational chain and Subsec. 8.4.3 provides the procedure necessary for its differentiation.

8.4.1 Structure of *Shaper*

The organisation of *Shaper* is shown in Fig. 8.1. Its C++ core functionalities are: parametric engine, data model and other plugins (marked in green layer). The other plugins work with the application only through event loop. The event loop is used by the sketch-solver (part of the parametric engine) to solve the sketch entities and constraints in real time. Moreover, the data model is based on the Open CASCADE Application Framework (OCAF) and automatically supports storage mechanism (in text, xml or binary format), transactions mechanism (undo/redo operations), nested transactions, etc. Its task is to provide data storage and

[2] `http://www.salome-platform.org/`

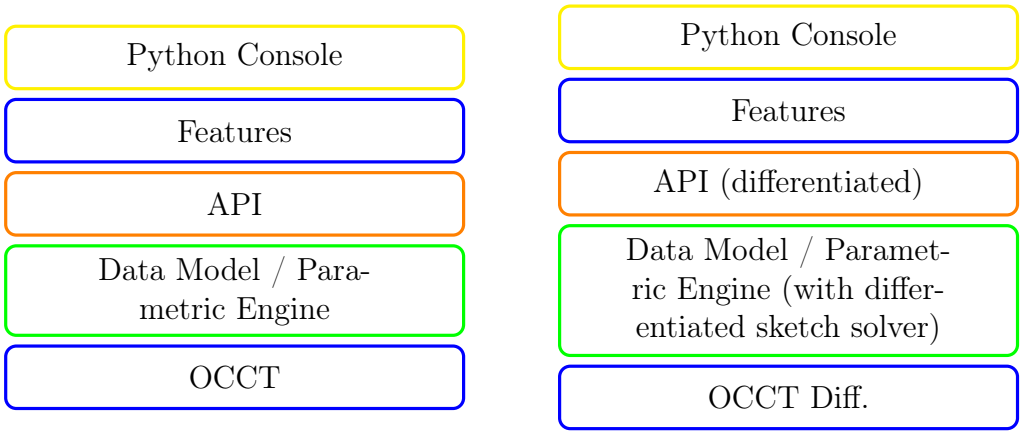


Figure 8.2: *Shaper* batch version (left) and differentiated batch version (right).

data manipulation functionalities. The data model of *Shaper* provides high-level API (Application Program Interface) that fully covers general and quite complex OCAF mechanisms. To develop an extra functionality (e.g. new feature), a developer will never use OCCT algorithms or OCAF directly but only the "API" classes. All the features/plugins are developed on the top of the API (in blue layer). They are implemented in C++ or python and they include all operations of the module and determine the functionality of the application. The yellow layer is the GUI (Graphical User Interface).

8.4.2 How to add Shaper to the computational chain?

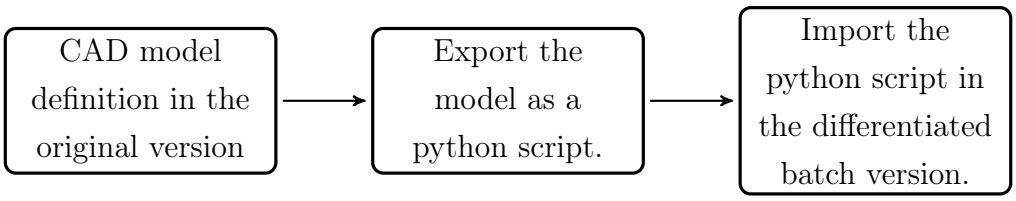
A possible solution to add *Shaper* to the computational chain is proposed in this subsection. Firstly, the author highlights that it is preferable that the CAD model of the test case is implemented in a not-differentiated environment (i.e. the sources used in the computational chain should not coincide with the one utilised to implement the parametrisation). This is mainly related to two reasons:

1. the GUI does not implement calculations to be differentiated and therefore can be excluded from the computational chain.

2. When utilised for designing a test case, the differentiated CAD system would be characterised by poor computational performance w.r.t. the non-differentiated version. The extension of all `real` variables to the active type `adouble` would increase the time needed by the CAD system to perform the calculations (e.g. time needed by the sketcher to solve the constraints).

A solution is the definition of a "batch" version of *Shaper* (a simplified scheme of the batch version is shown in Fig. 8.2). The batch version does not contain the

GUI (i.e. it only provides a python console interface), which allows to split the workflow of the designer into the following steps: (i) define the CAD model in the full version (i.e. by using the GUI), (ii) export the CAD model as a python script and (iii) import this script in the batch version:



The batch version of *Shaper* would be the one effectively used in the design chain. Therefore, this is the version that has to be differentiated.

8.4.3 Procedure to differentiate the "batch *Shaper*"

In order to differentiate the batch version of *Shaper*, the first step is the differentiation of the CAD kernel OCCT, which has been already achieved [105]. The sketcher (embedded inside the parametric engine) is, among the layers on the top of the OCCT kernel, the part where the majority of the calculations are carried out. It should be useful to extract it out of the parametric engine in order differentiate it as a stand alone project. After differentiating the sketcher, it has to be embedded into the parametric engine of the batch version of *Shaper*. This batch version is, then, based onto the differentiated version of OCCT such that a "differentiated batch version" is defined (shown in Fig. 8.2). The differentiation of the "batch *Shaper*" can be completed by differentiating the parts of the APIs and of the parametric engine where the other calculations are performed.

List of Figures

List of Tables

List of Code Listings

Appendix A

Parametrisation of pipes with OCCT.

```
// include all the classes used to parametrise the test case:

#include ...

Handle(Geom_Surface) geomFillSurface (Standard_Real radius)
{

// create a pathline curve (e.g. a B-spline curve)

// define the knots

TColStd_Array1OfReal pathKnots(1,4);
...

// define the Multiplicities

TColStd_Array1OfInteger pathMults(1,4);
pathMults(1)=3;
pathMults(2)=1;
pathMults(3)=1;
pathMults(4)=3;

// define the control points

TColgp_Array1OfPnt pathCP(1, 5);
pathCP(1) = gp_Pnt(0., 0, 0.);
pathCP(2) = gp_Pnt(0., 0, 10.);
pathCP(3) = gp_Pnt(0., 0, 20.);
pathCP(4) = gp_Pnt(0., 0, 30.);
pathCP(5) = gp_Pnt(0., 0, 40.);

// define the degree

Standard_Integer pathDegree = 2;

// ------- create the pathline ------------------

Handle(Geom_BSplineCurve) path =
  new Geom_BSplineCurve(pathCP, pathKnots, pathMults, pathDegree, Standard_False);

// -------- create the final surface --------------

Handle(Geom_Surface) pipeSurface = GeomFill_Pipe(path, radius).Surface();

return pipeSurface;

}
```

Code Listing A.1: Example (simplified) of a pipe constructed by using `GeomFill_Pipe` reference class.

```
// include all the classes used to parametrise the test case:

#include ...

Handle(Geom_Surface) geomSweepSurface (Standard_Real radius)
{

// create a law

const TColStd_Array1OfReal lawPoles;
const TColStd_Array1OfReal lawKnots;
const TColStd_Array1OfInteger lawMultiplicities;
const Standard_Integer lawDegree;

Law_BSpline lawBspline (lawPoles, lawKnots, lawMultiplicities, lawDegree, Standard_False);

Law_BSpFunc law(lawBspline, firstParam, lastParam);

// ------- create the pathline (see previous code listening) ------------------

Handle(Geom_BSplineCurve) path =
new Geom_BSplineCurve(pathCP, pathKnots, pathMults, pathDegree, Standard_False);

// create a profile

Handle(Geom_Circle) circProfile =
new Geom_Circle(gp::XOY(), 1.);

// Perpendicular section

Handle(GeomFill_SectionLaw)  sectionLaw  =
new GeomFill_EvolvedSection(circProfile, law);

Handle(GeomFill_LocationLaw) locationLaw =
new GeomFill_CurveAndTrihedron(new GeomFill_CorrectedFrenet);

Handle(GeomAdaptor_HCurve) pathAdt =
new GeomAdaptor_HCurve(path);

locationLaw->SetCurve(pathAdt);

// ------ Construct sweep -----------

// construction parameters

const GeomAbs_Shape continuity = GeomAbs_C2;
const int  maxDegree  = 25;

GeomFill_Sweep Sweep(locationLaw, 0);
Sweep.Build(sectionLaw, GeomFill_Location, continuity, maxDegree, maxSegment);

Handle(Geom_Surface) pipeSurface = Sweep.Surface();

return pipeSurface;

}
```

Code Listing A.2: Example (simplified) of a pipe constructed by using `GeomFill_Sweep` reference class.

```
// include all the classes used to parametrise the test case:

#include ...

TopoDS_Shape ThruSections ()
{

// Prepare "ThruSections"

const Standard_Boolean isSolid = Standard_False;
const Standard_Boolean isRuled = Standard_False;
const Standard_Real    pres3d  = 1.0e-06;

BRepOffsetAPI_ThruSections aGenerator(isSolid,isRuled,pres3d);

aGenerator.SetMaxDegree(5);

// define the slices

TopoDS_Wire wire1 = BRepBuilderAPI_MakeWire(aSection1);
TopoDS_Wire wirea2 = BRepBuilderAPI_MakeWire(aSection2);
TopoDS_Wire wireN = BRepBuilderAPI_MakeWire(aSectionN);


// Add section to the "ThruSections"

aGenerator.AddWire(Wire1);
aGenerator.AddWire(Wire2);
aGenerator.AddWire(WireN);

// compute the final shape

TopoDS_Shape tube = aGenerator.Shape();

return tube;

}
```

Code Listing A.3: Example (simplified) of a pipe constructed by using `BRepOffsetAPI_ThruSections` reference class.

Appendix B

Code listings for parametric CAD modeling of the two test cases.

```
// include all the classes used to parametrise the test case:

#include ...

TopoDS_Shape constructUbend (vector<Standard_Real> designParameters)
{

// ------------------------Stage 0: Initialise the design parameters--------------

for(int cnt = 0; cnt < designParameters.size(); cnt++) //nParams
   designParameters[cnt] <<= designParameters[cnt].getValue();

// ----------------------------------------

#define nbSlices 70

//-------------------------------------------------------------------------
// Stage 1: Construct laws from design parameters
//-------------------------------------------------------------------------

// define nb control points per laws of evolution

int numberCP = 8;

// ***************** z-coords of the laws ***********************


TColStd_Array1OfReal zCoords(1, numberCP);
zCoords(1)= 0.;
zCoords(numberCP)= 1.;

for (Standard_Integer zCoordParam = 2; zCoordParam < numberCP; ++zCoordParam)
    zCoords(zCoordParam)= (zCoordParam-1)/(double(numberCP)-1);

//CP1
TColgp_Array1OfPnt controlPoints1law(1, numberCP);
controlPoints1law(1) = gp_Pnt(-0.0375, 0.0375, zCoords(1));
controlPoints1law(2) = gp_Pnt(-0.0375, 0.0375, zCoords(2));
controlPoints1law(numberCP-1) = gp_Pnt(-0.0375, 0.0375, zCoords(numberCP-1));
controlPoints1law(numberCP) = gp_Pnt(-0.0375, 0.0375, zCoords(numberCP));


for(int indCP1 = 3; indCP1 < numberCP-1; ++indCP1)
controlPoints1law.SetValue(indCP1, gp_Pnt(designParameters[2*indCP1-2],
designParameters[2*indCP1-1], zCoords.Value(indCP1)));

Handle(Geom_BSplineCurve) BSplineCP1 = new Geom_BSplineCurve(controlPoints1law);

...

//-------------------------------------------------------------------------
// Stage 2: U-bend: B-Spline path for U-bend
//-------------------------------------------------------------------------

TColgp_Array1OfPnt pathPoles(1, 9);
Handle(Geom_BSplineCurve) path = new Geom_BSplineCurve(pathPoles);

//-------------------------------------------------------------------------
// Stage 3: Define the vector of sections
//-------------------------------------------------------------------------

int maxSlices = nbSlices;
TopoDS_Wire Wire[maxSlices];

GCPnts_UniformAbscissa pathDiscretizer (path, maxSlices, -1);
```

```
// Prepare skinning tool

const Standard_Boolean isSolid = Standard_False;
const Standard_Boolean isRuled = Standard_False;
const Standard_Real    pres3d  = 1.0e-06;

BRepOffsetAPI_ThruSections aGenerator(isSolid,isRuled,pres3d);
aGenerator.SetMaxDegree(5);

for(Standard_Integer i = 1; i <= maxSlices; ++i)
{

gp_Pnt P;
gp_Vec V;

path->D1(pathDiscretizer.Parameter(i), P, V);
gp_Dir d[i] = gp_Dir(V);

Standard_Real P_Length = GCPnts_AbscissaPoint::Length(path,
pathDiscretizer.Parameter(1), pathDiscretizer.Parameter(i));//length of the curve

Standard_Real Length_step = (P_Length/Length);

// -- Create the intersection plane
gp_Pnt P_path(0, 0, Length_step);

gp_Pln constructionPl(P_path, d2);
Handle(Geom_Plane) constructionPlane = new Geom_Plane (constructionPl);

//IP1: retrieve from law of CP1 the coordinates values for CP1 of the section
GeomAPI_IntCS IntCS1(BSplineCP1, constructionPlane);

// continue with other laws of evolutions (IntCS1/IntCS1/IntCS4):

...

TColgp_Array1OfPnt2d Bspline11(1, 4);
Bspline11(1) = gp_Pnt2d(IntCS1.Point(1).X(), IntCS1.Point(1).Y());
Bspline11(2) = gp_Pnt2d(IntCS2.Point(1).X(), IntCS2.Point(1).Y());
Bspline11(3) = gp_Pnt2d(IntCS3.Point(1).X(), IntCS3.Point(1).Y());
Bspline11(4) = gp_Pnt2d(IntCS4.Point(1).X(), IntCS4.Point(1).Y());
Handle(Geom2d_BSplineCurve) Bspline1 = new Geom2d_BSplineCurve(Bspline11);

// continue with other edges of the final section: Bspline2, Bspline3, Bspline4.

...

// Create wire

BRepBuilderAPI_MakeWire mkWire;
mkWire.Add(Bspline1);
mkWire.Add(Bspline2);
mkWire.Add(Bspline3);
mkWire.Add(Bspline4);
Wire[i] = mkWire.Wire();

// Add section to skinner
aGenerator.AddWire(Wire[i]);
}

TopoDS_Shape tube = aGenerator.Shape();

return tube;
}
```

Code Listing B.1: Implementation (simplified) of the function `constructUbend`.

```
// include all the classes used to parametrise the test case:

#include ...

TopoDS_Shape constructTUB (vector<Standard_Real> designParameters)
{

#define nbSlices 70

//-----------------------------------------------------------------------------
// Stage 1: Construct laws from design parameters
//-----------------------------------------------------------------------------

// define nb control points per laws of evolution

int numberCP = 8;

// *************** z-coords of the laws ***********************


TColStd_Array1OfReal zCoords(1, numberCP);
zCoords(1)= 0.;
zCoords(numberCP)= 1.;

for (Standard_Integer zCoordParam = 2; zCoordParam < numberCP; ++zCoordParam)
    zCoords(zCoordParam)= (zCoordParam-1)/(double(numberCP)-1);

//CP1
TColgp_Array1OfPnt camberCP1law(1, numberCP);
camberCP1law(1) = gp_Pnt(0., 0., zCoords(1));
camberCP1law(numberCP) = gp_Pnt(0., 0., zCoords(numberCP));


for(int indCP1 = 2; indCP1 < numberCP; ++indCP1)
camberCP1law.SetValue(indCP1, gp_Pnt(designParameters[2*indCP1-2],
designParameters[2*indCP1-1], zCoords.Value(indCP1)));

Handle(Geom_BSplineCurve) camberCP1 = new Geom_BSplineCurve(camberCP1law);

...

//-----------------------------------------------------------------------------
// Stage 2: Construction of the pathline
//-----------------------------------------------------------------------------

TColgp_Array1OfPnt pathPoles(1, 5);
Handle(Geom_BSplineCurve) path = new Geom_BSplineCurve(pathPoles);

// DISCRETIZE PATHLINE:

int maxSlices = nbSlices;

GCPnts_UniformAbscissa pathDiscretizer (path, maxSlices, -1);

// Prepare skinning tool

Standard_Boolean isSolid = Standard_False;
const Standard_Boolean isRuled = Standard_False;
const Standard_Real pres3d  = 1e-6;

BRepOffsetAPI_ThruSections aGenerator(isSolid,isRuled,pres3d);
aGenerator.SetMaxDegree(5);
```

```
for(Standard_Integer i = 0; i < maxSlices; ++i)
{

// construct the hosting planes for the slices

gp_Pnt P;
gp_Vec V;


path->D1(pathDiscretizer.Parameter(i+1), P, V);
d[i] = gp_Dir(V);

Standard_Real P_Length = GCPnts_AbscissaPoint::Length(path,
pathDiscretizer.Parameter(1), pathDiscretizer.Parameter(i+1));

Standard_Real Length_step = (P_Length/Length);

// -- Create the plane
gp_Pnt P_path(0, 0, Length_step);

gp_Pln constructionPl(P_path, d2);
Handle(Geom_Plane) constructionPlane = new Geom_Plane (constructionPl);

// retrieve from the laws of evolution the control points positions
//of camberline, suction/pressure BSplines.

// ex: cp number 1 of the camberline:
GeomAPI_IntCS IntCS11(camberCP1, constructionPlane);

// repeat this for all the other control points
//of the camberline/suction/pressure BSplines:

...

// construct camberline

TColgp_Array1OfPnt camberCP(1, 7);
camberCP(1) = gp_Pnt(IntCS11.Point(1).X(), IntCS11.Point(1).Y(), P.Z());

...

// construct the suction/pressure BSplines

Handle(Geom_BSplineCurve) pressureBS = new Geom_BSplineCurve(pressureBsplineCP);

Handle(Geom_BSplineCurve) suctionBS = new Geom_BSplineCurve(suctionBsplineCP);

TopoDS_Edge Edge1 = BRepBuilderAPI_MakeEdge(suctionBS);
TopoDS_Edge Edge2 = BRepBuilderAPI_MakeEdge(pressureBS);

BRepBuilderAPI_MakeWire mkWire;

mkWire.Add(Edge1);
mkWire.Add(Edge2);
Wire[i] = mkWire.Wire();

aGenerator.AddWire(Wire[i]);
}

TopoDS_Shape blade = aGenerator.Shape();

return blade;

}
```

Code Listing B.2: Implementation (simplified) of the function `constructTUB`.

Appendix C

Code listings for computing the area of intersection between two shapes in OCCT

```
// include all the classes used to compute the area of intersection:

#include ...

vector<Standard_Real> areaOfIntersection (TopoDS_Shape& blade, TopoDS_Shape& hole)
{
  TopoDS_Shape intersectionShapes = BRepAlgoAPI_Section(blade, hole);

  // store all the edges computed by the intersection between the blade and the hole
  // in the map "intersectingEdgeMap".

  TopTools_IndexedMapOfShape intersectingEdgeMap;
  TopExp::MapShapes(intersectionShapes, TopAbs_EDGE, intersectingEdgeMap);

  // assign the edges computed in the map to a list of shapes "TopTools_HSequenceOfShape".
  Handle(TopTools_HSequenceOfShape) edgesList = new TopTools_HSequenceOfShape;
  for(Standard_Integer k = 1; k <= edgesWireMap.Extent(); k++)
    edgesList.Append(edgesWireMap.FindKey(k));

  // get the intersecting sections by connecting the neighbors edges.

  Handle(TopTools_HSequenceOfShape) sectionList;
  ShapeAnalysis_FreeBounds freeBounds;
  freeBounds.ConnectEdgesToWires(edgesList, sectionList);

  // compute the area of the computed sections and store them in the vector
  // "intArea".

  ShapeAnalysis areaCalc[wireList.Length()];
  vector<Standard_Real> intArea;
  for(int i = 1; i <= sectionList.Length(); ++i)
  {
    TopoDS_Wire currentWire = TopoDS::Wire(sectionList.Value(i));
    intArea.push_back(areaCalc[i].ContourArea(currentWire));
  }

  return intArea;
}
```

Code Listing C.1: Simplified implementation of the method used to compute the area of intersection between the blade and the hole by using the topological intersection algorithm `BRepAlgoAPI_Section`.

```
// include all the classes used to compute the area of intersection:

#include ...

vector<Standard_Real> areaOfIntersection (TopoDS_Shape& blade, TopoDS_Shape& hole)
{
  // extract the surfaces from the blade and store them in the vector of surfaces
  // "bladeSurfaces".

  TopTools_IndexedMapOfShape bladeSurfaceMap;
  TopExp::MapShapes(blade, TopAbs_FACE, bladeSurfaceMap);

  vector<Geom_Surface> bladeSurfaces;
  for(int k = 1; k <= bladeSurfaceMap.Extent(); ++k)
    bladeSurfaces.push_back(bladeSurfaceMap.(k));

  // extract the surfaces from the hole and store them in the vector of surfaces
  // "holeSurfaces".

  TopTools_IndexedMapOfShape holeSurfaceMap;
  TopExp::MapShapes(hole, TopAbs_FACE, holeSurfaceMap);

  vector<Geom_Surface> holeSurfaces;

  for(int k = 1; k <= holeSurfaceMap.Extent(); ++k)
    holeSurfaces.push_back(holeSurfaceMap.(k));

  // now compute the intersection lines between the surfaces of both hole and blade.

  vector<TopoDS_Edge> interLinesEdges;

  for(int i = 0; i < bladeSurfaces.size(); ++i)
  {
    for(int j = 0; j < holeSurfaces.size(); ++j)
    {
      GeomAPI_IntSS IntSS(bladeSurfaces[i], holeSurfaces[j]);
      if(IntSS.IsDone())
      {
        IntSS.Perform();
        for(int i = 1; i <= IntSS.NbLines(); ++i)
              interLinesEdges.push_back(BRepBuilderAPI_MakeEdge(IntSS.Line(i)).Edge());
       }
     }
  }

  // now all the edges can be reordered in order to get the areas
  // of intersection between the blade and the hole and finally compute
  // the vector of "interArea", as done for the topological intersection algorithm.

  ...

}
```

Code Listing C.2: Simplified implementation of the method used to compute the area of intersection between the blade and the hole by using the geometrical intersection algorithm `GeomAPI_IntSS`.

```
// include all the classes used to compute the "polylineLength":

#include ...


Standard_Real polylineLength (Handle(Geom2d_Curve)& pressure2dcurve,
Handle(Geom2d_Curve)& suction2dcurve, vector<Standard_Real> holeCenterCoords)
{
  // Define the 2D section of the hole.
  Handle(Geom2d_Circle) circle = new Geom2d_Circle (gp_Pnt2d(holeCenterCoords[0],
  holeCenterCoords[1]), Radius); // Radius = 5.;

  // Initialise the 2D intersection algorithm between the circle and the pressure2dcurve
  // and store in the vector "interPnts" all the 2D intersection points.

  vector<gp_Pnt2d> interPnts;

  Geom2dAPI_InterCurveCurve intersectionCurvePressureSide (circle, pressure2dcurve);

  for(int k = 1; k <= intersectionCurvePressureSide.NbPoints(); ++k)
    interPnts.push_back(intersectionCurvePressureSide.Point(k));

  // Add to the vector "interPnts" also the points of intersection between
  // the circle and the suction curve.

 Geom2dAPI_InterCurveCurve intersectionCurveSuctionSide (circle, suction2dcurve);
  for(int k = 1; k <= intersectionCurveSuctionSide.NbPoints(); ++k)
    interPnts.push_back(intersectionCurveSuctionSide.Point(k));

  // Finally, compute the total length of the "polyline".

  Standard_Real polylineLength = 0.;
  for(int i = 1; i < interPnts.size(); ++i)
    polylineLength += interPnts[i].Distance(interPnts[i-1]);

  return polylineLength;
}
```

Code Listing C.3: Implementation (simplified) of the method used to compute the length of the polyline arising from the connection of the intersection points between the 2D circle (hole section) and the 2D section blade curves, intersection algorithm `Geom2dAPI_InterCurveCurve`. This method has been applied per every slice of the blade that intersects the hole.

Bibliography

[1] Ercan Oztemel and Samet Gursev. Literature review of industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*, 07 2018.

[2] Norbert Kroll, Klaus Becker, Herbert Rieger, and Frank Thiele. *MEGADESIGN and MegaOpt - German Initiatives for Aerodynamic Simulation and Optimization in Aircraft Design.* 01 2009.

[3] Shahrokh Shahpar. Challenges to overcome for routine usage of automatic optimisation in the propulsion industry. *Aeronautical Journal -New Series-*, 115, 05 2011.

[4] Joël Brezillon, Richard Dwight, and Jochen Wild. Numerical aerodynamic optimisation of 3D high-lift configurations. *ICAS Secretariat - 26th Congress of International Council of the Aeronautical Sciences 2008, ICAS 2008*, 4, 2008.

[5] X. Zhang. *CAD-based geometry parametrisation for shape optimisation using Non-uniform Rational B-splines.* PhD thesis, School of engineering and material science, Queen Mary University of London, 2017.

[6] P. Gaskell C. Gilkeson R. Hewson A. Keech H. Thompson A. Taherkhani, G. de Boer and V. Toropov. Aerodynamic drag reduction of emergency response vehicles. *Advances in Automobile Engineering*, 4, 2015.

[7] Selvakumar Ulaganathan, Pandiyarajan Raju, R. Mukesh, and K. Lingadurai. Influence of search algorithms on aerodynamic design optimisation of aircraft wings. volume 38, 04 2012.

[8] P. Gaskell R. Hewson V. Toropov A. Rezaienia A. R. Taherkhani, C. Gilkeson and H. Thompson. Aerodynamic CFD-based optimization of police car using bezier curves. *SAE International Journal of Materials and Manufacturing*, 10, 2017.

[9] Coletti F. Bulle J. Vanderwielen T. Verstraete, T. and T. Arts. Optimization of a U-bend for minimal pressure loss in internal cooling channels - part 1: Numerical method. *Journal of Turbomachinery*, 2011.

[10] P. Boulanger M. J. Garcia and S. Giraldo. CFD-based wing shape optimization through gradient-based method. In *Proceedings of the International Conference on Engineering Optimization*, 2008.

[11] Xingchen Zhang, Rejish Jesudasan, and Jens-Dominik Mueller. Adjoint-based aerodynamic optimisation of wing shape using non-uniform rational B-splines. In *International conference on evolutionary and deterministic methods for design optimization and control with application to industrial and societal problems (EUROGEN)*, 2017.

[12] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[13] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1989.

[14] Parviz Zadeh and Mohsen Sayadi. An efficient aerodynamic shape optimization of blended wing body uav using multi-fidelity models. *Chinese Journal of Aeronautics*, 31, 04 2018.

[15] M. Nemec D. W. Zingg and T. H. Pulliam. A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. *European Journal of Computational Mechanics/Revue Europeenne de Mécanique Numerique*, 17, 2008.

[16] Zelu Xu Zhoujie Lyu and JRRA Martins. Benchmarking optimization algorithms for wing aerodynamic design optimization. In *Proceedings of the 8th Interna- tional Conference on Computational Fluid Dynamics, Chengdu, Sichuan, China*, volume 11, 2014.

[17] Yin Yu, Zhoujie Lyu, Zelu Xu, and Joaquim R. R. A. Martins. On the influence of optimization algorithm and starting design on wing aerodynamic shape optimization. *Aerospace Science and Technology*, 75:183–199, 04/2018 2018.

[18] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial Mathematics, 2nd edition, nov 2008.

[19] Dheeraj Agarwal, Trevor Robinson, and Cecil Armstrong. A CAD based framework for optimizing performance while ensuring assembly fit. In *IC-SEE*, 5 2018.

[20] T. T. Robinson, C. G. Armstrong, H. S. Chua, C. Othmer, and T. Grahs. Optimizing parameterized CAD geometries using sensitivities based on adjoint functions. *Computer-Aided Design and Applications*, 9(3):253–268, 2012.

[21] Trevor Robinson, Cecil Armstrong, and Hung Chua. Determining the parametric effectiveness of a cad model. *Engineering with Computers*, 29, 01 2012.

[22] Jürgen Gräsel, Akin Keskin, Marius Swoboda, Hans Przewozny, and André Saxer. A full parametric model for turbomachinery blade design and optimisation. In *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2004.

[23] T. Verstraete. CADO: a computer aided design and optimization tool for turbomachinery applications. *In 2nd Int. Conf. on Engineering Optimization*, 2010.

[24] Ismael Sanchez Torreguitart and Tom Verstraete. Optimization of the LS89 axial turbine profile using a cad and adjoint based approach. *International Journal of Turbomachinery, Propulsion and Power*, 3, 2018.

[25] L. Mueller and T. Verstraete. Adjoint-based multi-point and multi-objective optimization of a turbocharger radial turbine. *International Journal of Turbomachinery, propulsion and power*, 2019.

[26] J. Dannenhoffer and J. Haimes. Design Sensitivity Calculations Directly on CAD-based Geometry. *53rd AIAA Aerospace Sciences Meeting, AIAA SciTech*, 2015. AIAA 2015-1370.

[27] D. Agarwal. *Shape optimization using feature-based CAD systems and adjoint methods, Queen's University Belfast*. PhD thesis, School of Mechanical and Aerospace Engineering, Queen's University of Belfast, 2018.

[28] S. Xu, J. Wolfram, and J. D. Müller. CAD-based Shape Optimisation with CFD using a Discrete Adjoint. *Int. J. Numer. Meth. Fluids, 74*, pages 153–168, 2013.

[29] A. Jameson. Aerodynamic design via control theory. In *Recent advances in computational fluid dynamics*, pages 377–401. Springer, 1989.

[30] Parry S.R. Sederberg, T. W. Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics*, 20:151–160, 1986.

[31] Taylor N.J. Masters, T.A. Review of aerofoil parameterisation methodsfor aerodynamic shape optimisation. *AIAA Aerospace Sciences Meeting*, 2015.

[32] R. M. Hicks and P. A. Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412, 1978.

[33] C. B. Allen D. J. Poole and T. C. S. Rendall. Control point-based aerodynamic shape optimization applied to aiaa adodg test cases. In *53rd AIAA Aerospace Sciences Meeting*, 2015.

[34] L. Piegl and W. Tiller. *The NURBS book*. Springer Science Business Media, 1995.

[35] R.L. Campbell. An approach to constrained aerodynamic design with application to aerofoils. *NASA, Technial Report*, 1992.

[36] Wu H., Yang S., and Liu F. Comparison of three geometric representations of airfoils for aerodynamic optimization. *16th AIAA Computational Fluid Dynamics Conference*, 2003.

[37] F. Christakopoulos. *Sensitivity computation and shape optimisation in aerodynamics using the adjoint methodology and Automatic Differentiation.* PhD thesis, School of engineering and material science, Queen Mary University of London, 2013.

[38] J. Vassberg and A. Jameson. Influence of shape parameterization on aerodynamic shape optimization. 05/2016, VKI lecture "Introduction to optimisation and Multidisciplinary design".

[39] A. Jameson and J. Vassberg. Studies of alternative numerical optimization methods applied to the brachistochrone problem. *Computational Fluid Dynamics Journal*, 3:281 – 296, 2000.

[40] Arthur Stück and Thomas Rung. Adjoint RANS with filtered shape derivatives for hydrodynamic optimisation. *Computers Fluids - COMPUT FLUIDS*, 47:22–32, 08 2011.

[41] *CAD-free vs CAD-based parametrisation method in adjoint based aerodynamic shape optimization.* Zenodo, September 2018.

[42] Orlando Soto, Rainald Lohner, and Chi Yang. A stabilized pseudo-shell approach for surface parametrization in cfd design problems. *Communications in Numerical Methods in Engineering*, 18:251 – 258, 04 2002.

[43] Seok-Hyung Bae and Byoung Choi. NURBS surface fitting using orthogonal coordinate transform for rapid product development. *Computer-Aided Design*, 34:683–690, 2002.

[44] Dwight R.P. Brezillon, J. Aerodynamic shape optimization using the discrete adjoint of the Navier-Stokes equations: Applications towards complex 3D configurations. In *CEAS/KATnet II Conference on Key Aerodynamic Technologies*, 2009.

[45] A.H. Barr. Global and local deformation of solid primitives. *Computer Graphics*, 18(3):21–30, 1984.

[46] J.A. Samareh. Aerodynamic shape optimization based on Free-Form deformation. *American Institute of Aeronautics and Astronautics Paper*, 2004–4630:1–13, 2004.

[47] Mouton S. Yamazaki, W. and G. Carrier. Efficient design optimisation by physics-based direct manipulation free-form deformation. *AIAA Conference*, 2008.

[48] Brezillon J. Ilic C. Widhalm, M. and T. Leicht. Investigation on adjoint based gradient computations for realistic 3D aero-optimisation. *13th AIAA/ISSMO Multidisciplinary Analysis Optimisation Conference*, 2010.

[49] Jamshid Samareh. Geometry and grid/mesh generation issues for CFD and CSM shape optimization. *Optimization and Engineering journal*, 6, 2005.

[50] M. E. Biancolini. Mesh morphing and smoothing by means of Radial Basis functions (RBF). In *Handbook of Research on Computational Science and Engineering: Theory and Practice: Theory and Practice*, 2011.

[51] Biancolini C. Costa E. Gattamelata D. Valentini P.P. Biancolini, M. E. Industrial application of the meshless morpher rbf morph to a motorbike windshield optimisation. In *the proceedings of The European Automotive Simulation Conference (EASC)*, 2009.

[52] Viola I.M. Riotte M. Biancolini, M.E. Sails trim optimisation using CFD and RBF mesh morphing. *Computers Fluids*, 2014.

[53] F. Gagliardi, K. Tsiakas, and K. Giannakoglou. *A Two–Step Mesh Adaptation Tool Based on RBF with Application to Turbomachinery Optimization Loops*, pages 127–141. 09 2019.

[54] Hadi Winarto and Arvind Sinha. Airfoil geometry parameterization through shape optimizer and computational fluid dynamics. In *AIAA aerospace Science meeting*, 2008.

[55] S. Eyi and K. D. Lee. Inverse aerofoil design using the navier-stokes equations. *Engineering Optimisation*, 24(8):245–262, 1997.

[56] Sasaki D. Obayashi S. Kim, H.J. and K. Nakahashi. Aerodynamic optimisation of supersonic transport wing using unstructured adjoint method. *AIAA Journal*, 39(6):1011–1020, 2001.

[57] Alonso J.J. Kim, S. and A. Jameson. Design optimisation of high-lift configurations using a viscous continuous adjoint method. *AIAA Conference*, 2002.

[58] Kim H-J. Matsushima K. Nakahashi K. Nakayama, H. and K. Takenaka. Aerodynamic optimisation of multi-element aerofoil. *AIAA Conference*, 2006.

[59] J. Elliott and J. Peraire. Practical 3D aerodynamic design and optimisation using unstructured meshes. *AIAA Conference*, 1996.

[60] Hyoung-Jin Kim and Kazuhiro Nakahashi. Unstructured Adjoint Method for Navier-Stokes Equations. *JSME International Journal Series B*, 48(2):202–207, January 2005.

[61] Kwon J.H. Sung, C.H. Aerodynamic design optimisation using the navier-stokes and adjoint equations. *AIAA*, 2001.

[62] C. B. Allen D. J. Poole and T. C. S. Rendall. High-fidelity aerodynamic shape optimization using efficient orthogonal modal design variables with a constrained global optimizer. *Computer and Fluids*, 143:1–15, 2017.

[63] D. J. Poole C. B. Allen and T. C. S. Rendall. Wing aerodynamic optimization using efficient mathematically-extracted modal design variables. *Optimization and Engineering*, 19(2):453–477, 2018.

[64] J.-Y. Trépanier J. Lépine and F. Pépin. Wing aerodynamic design using an optimized NURBS geometrical representation. In *38th Aerospace Sciences Meeting and Exhibit*, 2000.

[65] J.-Y. Trépanier J. Lépine, F. Guibault and F. Pépin. Optimized nonuniform rational b-spline geometrical representation for aerodynamic design of wings. *AIAA journal*, 39(11):2033, 2001.

[66] Simon Painchaud-Ouellet, Christophe Tribes, Jean-Yves Trépanier, and Dominique Pelletier. Airfoil shape optimization using a nonuniform rational B-splines parametrization under thickness constraint. *Aiaa Journal - AIAA J*, 44:2170–2178, 10 2006.

[67] C. R. Barrios. Thinking parametric design: introducing parametric. *Design Studies*, 27(3):309–324, 2009.

[68] JJ. Shah. Assessment of features technology. *Computer-Aided Design*, 23(5):331–343, 1991.

[69] D. Roller. An approach to computer-aided parametric design. *Computer-Aided Design*, 23(5):385–391, 1991.

[70] Mendgen R. Anderl, R. Modelling with constraints: theoretical foundation and application. *Computer-Aided Design*, 28(3):155–168, 1996.

[71] Zhang D.L. Yuen-M.M.F. Liu, Y.J. A survey on CAD methods in 3D garment design. *Computer-Aided Design*, 61(6):576–593, 2010.

[72] Tong T. Ma, Y.S. Associative feature modeling for concurrent engineering integration. *Computers in industry*, 51(1):51–71, 2003.

[73] McMahon Salehi, V. Development and application of an integrated approach for parametric associative CAD design in an industrial context. *Computer-Aided Design and Applications*, 8(2):225–236, 2011.

[74] Marian Nemec, Michael Aftosmis, and Tom Pulliam. CAD-Based aerodynamic design of complex configurations using a cartesian method. *42nd AIAA Aerospace Sciences Meeting and Exhibit*, 01 2004.

[75] Ilias Vasilopoulos, Peter Flassig, Marcus Meyer, and Kyra Reich. Aerodynamic Optimization of the TurboLab Stator: A Comparative Study between Conventional and Adjoint-based approaches. In *International Conference on Numerical Optimisation Methods for Engineering Design (NOED)*, 2016.

[76] R. Haimes. CAPRI (computational analysis programming interface): A solid modeling based infra-structure for engineering analysis and design simulations. In *NASA technical reports*, 1998.

[77] Daniel Fudge, David Zingg, and Robert Haimes. A cad-free and a cad-based geometry control system for aerodynamic shape optimization. *American Institute of Aeronautics and Astronautics CP 05-0451*, 01 2005.

[78] Dean Bryson, Robert Haimes, and John Dannenhoffer. Toward the realization of a highly integrated, multidisciplinary, multifidelity design environment. 01 2019.

[79] Robert Haimes and John Dannenhoffer. EGADSlite: A lightweight geometry kernel for HPC. In *56th AIAA Aerospace Sciences Meeting*, 2018.

[80] John Dannenhoffer and Robert Haimes. Using design-parameter sensitivities in adjoint-based design environments. In *55th AIAA Aerospace Sciences Meeting*, 2017.

[81] Robinson T.T. Armstrong, C.G. and H.S. Chua. Parametric effectiveness of cad models and strategies for its improvement. In *8th World Congress on Structural and Multidisciplinary Optimisation*, 2009.

[82] Dheeraj Agarwal, Trevor T. Robinson, Cecil G. Armstrong, Simão Marques, Ilias Vasilopoulos, and Marcus Meyer. Parametric design velocity computation for cad-based design optimization using adjoint methods. *Engineering with Computers*, 34(2):225–239, Apr 2018.

[83] Dheeraj Agarwal, Christos Kapellos, Trevor Robinson, and Cecil Armstrong. Using parametric effectiveness for efficient CAD-based adjoint optimization. *Computer-Aided Design and Applications*, 16, 05 2018.

[84] L. Piegl. On NURBS: A survey. *Computer Graphics and Applications, IEEE*, 11(1):55–71, 1991.

[85] O. Mykhaskiv, M. Banovic, S. Auriemma, P. Mohanamuraly, A. Walther, H. Legrand, and J.-D. Müller. NURBS-based and parametric-based shape optimisation with dierentiated CAD kernel. *Computer-Aided Design and Applications*, 2017. `https://qmro.qmul.ac.uk/xmlui/handle/123456789/25266`.

[86] Mario Martin, Esther Andres, M. Widhalm, P. Bitrián, and Carlos Lozano. NURBS-based aerodynamic shape design optimization with the dlr tau code. *Proceedings of the Institution of Mechanical Engineers Part G Journal of Aerospace Engineering*, 01 2011.

[87] Esther Andres, P. Bitrián, Carlos Lozano, Mario Martin, and M. Widhalm. Preliminary comparisons between two CAD-based aerodynamic shape optimization approaches using adjoint methods for fast gradient computation. In *EngOpt2010, 2nd International Conference on Engineering Optimization*, 2010.

[88] Anas Bentamy, Francois Guibault, and Jean-Yves Trépanier. Aerodynamic optimization of a realistic aircraft wing. In *43rd AIAA Aerospace Sciences Meeting and Exhibit - Meeting Papers*, 2005.

[89] Jens-Dominik Müller, Xingchen Zhang, Siamak Akbarzadeh, and Yang Wang. Geometric continuity constraints of automatically derived parametrisations in cad-based shape optimisation. *International Journal of Computational Fluid Dynamics*, 33(6-7):272–288, 2019.

[90] Orest Mykhaskiv, Pavanakumar Mohanamuraly, Jens-Dominik Mueller, Shenren Xu, and Sebastian Timme. Cad-based shape optimisation of the nasa crm wing-body intersection using differentiated cad-kernel. In *35th AIAA Applied Aerodynamics Conference*, 06 2017.

[91] Peter Sturdza Joaquim RRA Martins and Juan J Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 3:245–262, 2003.

[92] A. Walther and Vogel O. Griewank, A. *ADOL-C: Automatic Differentiation Using Operator Overloading in C++*. Combinatorial scientific computing. PAMM, 2003.

[93] A. Walther and A. Griewank. *Getting Started with ADOL-C*, pages 181–202. Combinatorial scientific computing. 2012.

[94] L. Hascoët and V. Pascual. The tapenade automatic differentiation tool: Principles, model, and specification. *ACM Transactions On Mathematical Software*, 39(3), 2013.

[95] R. Giering. Tangent linear and adjoint model compiler, users manual. *Center for Global Change Sciences, Department of Earth, Atmospheric, and Planetary Science, MIT, Cambridge, MA, December 1997. Unpublished.*

[96] Jieqiu Chen, Paul Hovland, Todd Munson, and Jean Utke. An integer programming approach to optimal derivative accumulation. In Shaun Forth, Paul Hovland, Eric Phipps, Jean Utke, and Andrea Walther, editors, *Recent Advances in Algorithmic Differentiation*, volume 87 of *Lecture Notes in Computational Science and Engineering*, pages 221–231. Springer, Berlin, 2012.

[97] Max Sagebaum, Tim Albring, and Nicolas Gauger. High-Performance derivative computations using CoDiPack. *ACM Transactions on Mathematical Software*, 45, 09 2017.

[98] Claus Bendtsen and Ole Stauning. FADBAD, a flexible C++ package for automatic differentiation - using the forward and backward methods. *Technical report, Department of mathematical modeling, Technical University of Denmark*, 1996.

[99] Bradley Bell and James Burke. *Algorithmic Differentiation of Implicit Functions and Optimal Values*, volume 64, pages 67–77. 08 2008.

[100] O. Vogel and Andreas Griewank. Direct gear tooth contact analysis for hypoid bevel gears. *Computer Methods in Applied Mechanics and Engineering*, 191, 09 2001.

[101] Klaus Röbenack. Automatic differentiation and nonlinear controller design by exact linearization. *Future Generation Computer Systems*, 21(8):1372 – 1379, 2005.

[102] Eckhard Arnold, H. Linke, and R. Franke. Optimal control application for operational water management of a canal system. *IFAC Proceedings Volumes*, 31:39–44, 07 1998.

[103] Gundolf Haase, Ulrich Langer, Ewald Lindner, and W. Muhlhuber. Optimal sizing using automatic differentiation. *ISNM International Series of Numerical Mathematics*, 2001.

[104] Annual statistics from OCCT users. *Open CASCADE internal report*, 2018.

[105] M. Banovic, O. Mykhaskiv, S. Auriemma., A. Walther, H. Legrand, and J.-D. Müller. Automatic Differentiation of Open CASCADE Technology CAD System and its coupling with an Adjoint CFD Solver. *Optimisation methods and Software*, 2017. `http://www.optimization-online.org/DB_HTML/2017/03/5931.html`.

[106] J. Burman. *Geometry Parameterisation and Response Surface-Based Shape Optimisation of Aero-Engine Compressors.* PhD thesis, Department of Applied Physics and Mechanical Engineering,Divisions of Fluid Mechanics, Lulea University of Technology, 2003.

[107] P. Castonguay and Nadarajah Silva K. Effect of shape parametrization on aerodynamic shape optimization. *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007.

[108] D.A. Masters, N. J. Taylor, T.C.S. Rendall, C.B. Allen, and D.J. Poole. A geometric comparison of aerofoil shape parameterisation methods. *AIAA journal*, 2017. http://arc.aiaa.org/doi/abs/10.2514/1.J054943.

[109] Juned Ahamad R. A. Channiwala S.A. Salunke, N.P. Airfoil parameterization techniques: A review. *American Journal of Mechanical Engineering*, 4(2), 2014.

[110] Alfredo Arias-Montano, Carlos Coello, and Efrén Mezura-Montes. Evolutionary algorithms applied to multi-objective aerodynamic shape optimization. *Studies in Computational Intelligence*, 356:211–240, 1970.

[111] Pierluigi Della Vecchia, Elia Daniele, and Egidio DAmato. An airfoil shape optimization technique coupling PARSEC parameterization and evolutionary algorithm. *Aerospace Science and Technology*, 32(1):103 – 110, 2014.

[112] Peng F. Chen, J. Approximate spline of g2-continuity on a generalized hyperbolic paraboloid. *Journal of Computational and Applied Mathematics*, pages 99–117, 1013.

[113] W. Tiller. Rational B-splines for curve and surface representation. *Computer Graphics and Applications*, 6(3):61–69, 1983.

[114] Bo P. Liu Y. Wang W. Zheng, W. Fast B-spline curve fitting by l-bfgs. *Computer Aided Geometric Design*, pages 448–462, 2012.

[115] Wright S.J. Nocedal, J. Numerical optimization. *Springer-Verlag*, 1999.

[116] T. Verstraete. *Multidisciplinary turbomachinery component optimization considering performance, stress, and internal heat transfer*. PhD thesis, Von Karman Institute, 2008.

[117] Rajaratnam Shanthini. *Thermodynamics for Beginners - Chapter 12 THERMODYNAMIC ANALYSES OF POWER PLANTS*, pages 273 – 304. 01 2006.

[118] S.M. Chang, J.A.C. Humphrey, and A. Modavi. Turbulent flow in a strongly curved u-bend and downstream tangent of square cross-section. *PCH. Physicochemical hydrodynamics*, 4:243–269, 01 1983.

[119] S. Cheah, H. Iacovides, D. Jackson, H. Ji, and Brian Launder. LDA investigation of the flow development through rotating u-ducts. *Journal of Turbomachinery-transactions of The Asme - J TURBOMACH-T ASME*, 118, 07 1996.

[120] L. Mueller T. Verstraete and J.-D. Mueller. Adjoint-based design optimisation of an internal cooling channel u-bend for minimised pressure losses. *International Journal of Turbomachinery, Propulsion and Power*, 2, 2017.

[121] Hervè Legrand. Curve/surface by smoothing N points/lines: internal specifications. In *OCC internal report*.

[122] Dheeraj Agarwal, Trevor Robinson, Cecil Armstrong, and Christos Kapellos. Enhancing CAD-based shape optimisation by automatically updating the CAD model's parameterisation. *Structural and Multidisciplinary Optimization*, 59, 11 2018.

[123] Ping He, Charles Mader, Joaquim Martins, and Kevin Maki. Aerothermal optimization of a ribbed u-bend cooling channel using the adjoint method. *International Journal of Heat and Mass Transfer*, 06 2019.

[124] M. Banovic. *Automatic Differentiation of Open CASCADE Technology CAD System.* PhD thesis, Department of mathematics, University of Paderborn, 2019.

[125] O. Pirennau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64:97–110, 1974.

[126] J.D. Mueller. A review of adjoint method. In *Internal report, School of engineering and material science, Queen Mary University of London*, 2007.

[127] M. B. Giles, M. C. Duta, J. D. Müller, and N. A. Pierce. Algorithm developments for discrete adjoint methods. *AIAA journal*, 41(2):198–205, 2003.

[128] P. Cusdin and J. D. Müller. On the performance of discrete adjoint CFD codes using automatic differentiation. *Numerical Methods in Fluids*, 2005.

[129] J.M. Malé B. Mohammadi and N. Rostaing-Schmidt. Automatic differentiation in direct and reverse modes: Application to optimum shapes design in fluid mechanics. *In Martin Berz, Christian H. Bischof, George F. Corliss, and Andreas Griewank, editors, Computational Differentiation: Techniques, Applications, and Tools*, page 309–318, 1996.

[130] Dervieux A. Koobus B. Hascoët L. Courty, F. Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation. *Optimization Methods and Software*, 37(2):185 – 191, 1999.

[131] B. Mohammadi. Optimal shape design, reverse mode of automatic differentiation and turbulence. 1997.

[132] Mihai Duta and Michael Giles. The use of automatic differentiation for adjoint CFD codes. In *Post SAROD Workshop*, 2005.

[133] Jones D. Müller J. D. Christakoupoulos, F. Pseudo-timestepping and validation for automatic differentiation derived code. *Computers and Fluids*, 46(1):174 – 179, 2011.

[134] S. Xu J. Hückelheim J.-D. Mueller, P. Mohanamuraly and M. Gugala. STAMPS: a finite-volume solver framework for adjoint codes derived with source-transformation ad. *AIAA-CP XX-2018,*, page submitted, 2018.

[135] M. Gugala. *Output-based mesh adaptation using geometric multi-grid for error estimation.* PhD thesis, School of Engineering and Materials Science, Queen Mary University of London, 2018.

[136] James J McGuirk Andreas Haselbacher and Gary J Page. Finite volume discretization aspects for viscous flows on mixed unstructured grids. *AIAA journal*, 37(2):177–184, 1999.

[137] Boris Diskin and James L Thomas. Comparison of node-centered and cell-centered unstructured finite-volume discretizations: inviscid fluxes. *AIAA journal*, 49(4):836–854, 2011.

[138] Steven R Allmaras and Forrester T. Johnson. Comparison of node-centered and cell-centered unstructured finite-volume discretizations: inviscid fluxes. In *Seventh international conference on computational fluid dynamics (ICCFD7)*, pages 1–11, 2012.

[139] B. Christianson. Reverse accumulation and implict functions. *Optimization Methods and Software*, 9(4):307 –322, 1998.

[140] Jamshid A Samareh. A survey of shape parameterization techniques. *NASA Conference Publication*, 41(2):333–344, 1999.

[141] PG A. Cizmas and J. Gargoloff. Mesh generation and deformation algorithm for aeroelasticity simulations. *Journal of Aircraft*, 45(3):1062 – 1066, 2008.

[142] Van der Schoot M.S. Bijl H. De Boer, A. Mesh deformation based on radial basis function interpolation. *Computer and Structures*, 85(11):784–795, 2007.

[143] J. Witteveen and H. Bijl. Explicit mesh deformation using inverse distance weighting interpolation. *19th AIAA Computational Fluid Dynamics Conference*, 2009.

[144] Filippo Coletti, Tom Verstraete, Jérémy Bulle, Timothée Van der Wielen, Nicolas Van den Berge, and Tony Arts. Optimization of a U-Bend for Minimal Pressure Loss in Internal Cooling Channels, Part II: Experimental Validation. *Journal of Turbomachinery*, 5, 2013.

[145] F. Christakopoulos, D. Jones, and J.-D. Müller. Pseudo-timestepping and validation for automatic differentiation derived code. *Computers and Fluids*, 46:174–179, 2011.

[146] Rejish Jesudasan, Xingchen Zhang, and Jens-Dominik Mueller. Adjoint optimisation of internal turbine cooling channel using nurbs-based automatic and adaptive parametrisation method. In *ASME Gas Turbine Conference*, 2017.

[147] J. Bulle T. Van der Wielen N. Van den Berge F. Coletti, T. Verstraete and T. Arts. Optimization of a u-bend for minimal pressure loss in internal cooling channels - part ii: Experimental validation. *Journal of Turbomachinery*, 135, 2013.

[148] Rejish Jesudasan, Xingchen Zhang, Mateusz Gugala, and Jens-Dominik Müller. CAD-free vs CAD-based parametrisation method in adjoint-based aerodynamic shape optimization. In *ECCOMAS Congress 2016*, 2016.

[149] Jens-Dominik Müller R. Jesudasan. CAD-based parametrisation framework for aerodynamic shape optimisation using adjoint sensitivities. *TO BE SUBMITTED*, 2021.

[150] J. D. Mueller. Personal communications. 2020.

[151] J.-D. Müller, P. Mohanamuraly, S. Xu, J. Hückelheim, and M. Gugala. STAMPS: a finite-volume solver framework for adjoint codes derived with source-transformation AD. *AIAA-CP XX-2018*, 2018.

[152] O. Mykhaskiv. Optimal shape design with automatically differentiated CAD parametrisations. *Submitted in partial fulfillment of the requirements of the Degree of Doctor of Philosophy, Queen Mary University of London*, 2019.

[153] Dieter Peitsch Ilias Vasilopoulos Jan Mihalyovics, Christian Brueck and Marcus Meyer. CAD-based aerodynamic optimization of a compressor stator using conventional and adjoint-driven approaches. In *Turbomachinery Technical Conference and Exposition*, 2018.

[154] Salvatore Auriemma, Mladen Banovic, Orest Mykhaskiv, Jens-Dominik Mueller, and Andrea Walther. Applications of differentiated CAD kernel in gradient-based aerodynamic shape optimisation. In *AIAA AIAA Propulsion and Energy Forum and Exposition*, 2018.

[155] D. Kraft. A software package for sequential quadratic programming. *Technical Report, DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen*, 1988.

[156] OCCT: intersection algorithm. In *OCC internal report.*

[157] E. Papoutsis-Kiachagias and K. Giannakoglou. Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. *Archives of Computational Methods in Engineering*, 2014.

[158] Projections with OCCT. In *OCC internal communication, 2019*.

[159] A geometric modeler for SALOME. In *OCC internal report*.